Master's Thesis

Researching of the deep neural network for amber gemstone classification

---

Author: Ramiro Saito Castro Rios

Supervisor: Arunas Lipnickas

Polytechnic University of Catalonia, UPC

Barcelona School of Industrial Engineering, ETSEIB

Kaunas University of Technology, KTU

Master's degree in Automatic Control and Robotics

June 1th, 2018

Abstract

This project is based on the researching of the deep neural network for the classification of amber gemstone seen as a great opportunity to expand the range of application of the well known deep learning, which have been used more and more in different fields. First, it is explained the theoretical concepts about machine learning and deep learning, besides a little comparison between them in order to for the reader to make the thesis more understandable. It is also introduced the different machine learning classifiers and the different methods of deep learning which have been used during the presented work. After having the basic concepts it will be explained the different methods that have been studied throughout the thesis to find the best possible accuracy in the training, validation and testing of the data set. Finally, it will be showcased the different results obtained followed by a short explanation per each of them.

In this work the main technique that will be used for the researching is the method of transfer learning. Due to the lack of huge amounts of database regarding the amber stones, this technique can be a lot profitable to achieve the best accuracy. Nevertheless other methods like training from scratch will be tested; in which it will be used YOLO (You Only Looks Once). There is plenty of developed architectures that can be used for transfer learning, but only AlexNet will be used, since is one of the most used. The main tool for performing all the training and testing is MATLAB, which is quite suitable for processing images and with the latest version now it is able to run deep neural network with ease.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
DT	Decision Tree
$k-NN$	k- nearest neighbours
HSV	Hue, Saturation and Value
HSI	Hue, Saturation and Intensity
mAP	Mean Average Precision
NB	Naive Bayes
NN	Neural Network
PDT	Pruned Decision Tree
QDA	Quadratic Discriminant Analysis
RBF	Radial Basis Function
RNN	Recurrent Convolutional Neural Network
$RELU$	REctified Linear Unit
ROI	Region Of Interest
SVM	Support Vector Machine
$YOLO$	You Only Looks Once

Chapter 1

Introduction

Amber stone is a fossilized resin of vegetable origin used to create impressive jewellery, souvenirs, mosaics and fine art crafts. This stone has always being appreciated since the neolithic times and is valued as gemstone. These are mostly found in the baltic sea and nowadays, the majority of amber stones are extracted from Kaliningrad Oblast in Russia; around 90% of stones. They are collected by hand, dredging or diving for a later classification. The classification of ambers is a hard job carried out by art craft industry experts, so that the necessity of performing an automated system that can sort them by colour and shapes arised. This presented work is one of the few works dedicated to the automated classification of amber stones, since only a few researches dedicated their time in this issue due to the difficulty of the task. A fully automated industrial sorting line based on machine vision is the best solution for amber classification and it can suppose an incredibly reduction of the time consuming which in turns can leads to a bigger production of amber-based products.

1.1 Motivation and objectives of the project

The main motivation of the presented thesis is related to the use of the deep learning technique for the classification of a complicated database based on texture; amber

stone. By performing and analysing different methodologies of deep learning for the pursuit of the best accuracy in this type of data, it is expected to contribute in the researching world of this area, which arose with the arrival of what is called Big Data.

The second main motivation is to demonstrate whether by using deep learning, which is day by day being tested in different fields, is able to improve the classification of amber gemstone regarding to the last researching performed. Thereby, following such motivations a list of objectives are set:

- **First Objective**: Knowing the theoretical concepts needed to understand the different methodologies applied for the researching in classification of amber stones.
- **Second Objective**: Knowing how is divided the database and how which kind of pre-processing have been carried out to this database to go deeper in the researching.
- **Third Objective**: Once set the base of the problem, the third objective is to give a deep explanation of all methodologies applied in order to achieve different performance in the classification.
- **Fourth Objective**: The last objective is showing all the results obtained from the different methods and give to the reader a short explanation about the reason why one performance is better than others.

Chapter 2

State of the art of amber stone classification

Amber stone classification is one task that just a few researchers have started to think of due to the necessity of setting an automation system that classify them by colour or shapes. Classify ambers is a hard work that only expert can do it and is too much time consuming. During these last years some types of classifications have been carried out like classification by colour, by shapes and by colour/shapes. Nevertheless, this task is similar as classify objects by texture.

One of the related works done is a classification in weld radiographic images using geometric and texture features [3] for detecting defects such as worm holes, porosity, linear slag inclusion, gas pores among others. In here, a set of 43 descriptors corresponding to texture measurements and geometrical features is extracted for each segmented object and given as input to a classifier. The classifier tested are Support Vector Machine, Neural Network and k-NN and a cross validation is used for testing.

Another researching work about texture classification is [7], where it is tried to classify food grains using different color models and haralick features. In that paper

it is used color models such as HSV, HSI, L*a*b and YCbCr and only by combining color and texture features without any preprocessing it is performed the training. For that it is used K-NN and minimum distance classifier trained with local and global features.

Early related work to the amber stone classification it is found in **[8]**. Here it is done the classification of amber by similarity to basic geometric shapes. It is used k-medoids and k-means clustering algorithms for labelling one of the geometric shapes: circle, ellipse, oval, triangle, rectangle, rhombus, trapezium and trapezoid. However, clusters were labelled by experts and were used to a late classification. In this case, decision tree classifier was trained by using Centroid Distance Function (CDF) feature.

Another work is found in **[10]** called *"Automatic amber gemstones identification by color and shape visual properties"*. In that paper it is proposed an amber classification by color and shape. There are 30 classes from which 20 of those are used for shape identification and all 30 classes for color identification. On the one hand, for color identification it is used concentric rings (ROI) segmentation in order to have more statistical information from each sample. As a descriptor it is taken the HSV and grayscale color features due to the fast performance. From each ROI it is extracted four statistical measures: Standard deviation, Kurtosis and Skewness. The classifiers tested with those features are Quadratic Discriminant Analysis (QDA), K Nearest Neighbours (KNN), Radial Basis Function (RBF), Decision Tree (DT) and Pruned Decision Tree (PDT); finally majority vote criteria is implemented for identification. The best classification result with only a simple segmentation is obtained from the Pruned Decision Tree classifier with a **69.29%** but then by introducing concentric rings increased up to 10% more, so **79%**. The final accuracy is got after few more experiments with a limited size of ring areas (max. 400 pixels per ring) and by forming a committee of Decision Tree classifiers (16DT) with Half & Half

method, increasing 2-3% more. Hence, the final accuracy is **81.50%**

On the other hand, for shape identification it is used the Centroid Distance Function (CDF); a vector expressed by the distance of the landmark points from the centroid. This feature is chosen because is invariant to translation and orientation. The classifiers used are the same as for color identification with the exception of QDA which has been replaced by Naive Bayes (NB). The best accuracy was acquired by K-NN. However as it is too slow, DT was used instead getting an accuracy of **72.10%**.

The latest researching on amber gemstones classification are in **[9]** from the past year, called *"Amber Gemstone sorting by Colour"*. In such paper, it is presented a final accuracy of **88.21%**. At first sized concentric rings were suggested to calculate the first order statistical features from HSV color space. With this features and with Decision Tree classifier and Half and Half method it is got an accuracy of **77.90%**. But later on, by using rejection threshold technique it is seen a clearly improvement obtaining a final accuracy of 88.21%.

Chapter 3

Theoretical framework

In this chapter of the theoretical framework it is intended to explain all the theory related with the work such as concepts about machine learning, deep learning among others to help the reader refresh concepts about what is known. However, not only theoretical concepts are explained in this chapter but it will be explained the type of data that will be analysed as a researching; from how it was obtained until the different modifications that it has been carried out needed for the researching.

3.1 Machine Learning

Machine learning is the ability of a computer (machine) to "learn" progressively given a certain data by assessing the performance on a specific task. The concept of machine learning is evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning look into the construction of algorithms that can learn from and make prediction on data by building a model from sample inputs. Basically, it is employed in a range of computing tasks where static programming are quite difficult to design.

The best way to make machines that can learn from data is to use the tools of probability theory. Probability theory can be applied to any problem involving un-

certainty which can come in many forms like noise from sensors measurements or blurry images. In the same manner, probability is also used to assess which class a certain data belongs to.

In order to reach the solution of many prediction and classification problems there exist three types of machine learning: supervised learning, unsupervised learning and reinforcement learning. Each of them deal with different problems.

- **Supervised Learning**: This type of machine learning is mostly used when you have the training data and also the targets. It is mean, when you know which class each sample belongs to. Hence, the machine is trained from training set and target vector. Then is tested with new data in order to evaluate the performance itself.

- **Unsupervised Learning**: If it is the case that it is got a certain amount of data which contains information about different patterns and it is not known the desired output for each input, then this kind of machine learning is used. This works in a way creating clusters and density estimation for the given data, whose goal is to find "interesting patterns" on it. In this case there is no straightforward way to evaluate the accuracy of the structure that is produced by the algorithm.

- **Reinforcement Learning**: The idea of this type of machine learning is different from the above mentioned. In this case, it is tried to discover the solution by trial and error, about how to act or behave when given occasional reward or punishment signals. This is inspired about how software ***agents*** ought to take actions in an ***environment*** so as to maximise some cumulative ***reward***

3.1.1 SVM: Support Vector Machines

Support vector machine is a supervised machine learning algorithm which can be used for classification and regression analysis, although it is mostly used for classi-

fication purpose. Nowadays, is one of the most widely used clustering algorithms in industrial applications. SVM can perform linear classification and efficiently non-linear classification by means of what is called the "kernel trick", which is introducing the initial space of the non linear data in another one with higher features dimension, where data can be linearly separable.

Identify the right hyperplane is the main problem of any classification problem. **Figure 3.1** attempts to explain the main problem of classification. In the left image is difficult to know which line separates better the two classes, although it seems to be the line B. At the right image, with those 3 lines what it is needed to do is to maximise the distances between nearest data point and hyper-plane to decide the right plane. This distance is called **Margin**.

Basically, what support vector machines does is to find the optimal hyperplane or set of optimal hyperplanes in a high or infinite dimensional space, such that the margin is maximum: $max(dist(X_{min}) = max(\frac{1}{|w|})$. SVM has the feature to ignore outliers data and find the hyper plane that has maximum margin. For the optimization only the points that are nearest to the hyperplane are taken and they are called support vectors, since from them depends the classification of the rest of the data. **Figure 3.1** shows an example of hyperplane found by SVM, so it is seen is the optimal line that separates both classes.

In the case of non linear classifiers, SVM can manage efficiently by using the kernel trick. Kernels are functions which takes low dimensional input space and transform it to a higher dimensional space. For example, in **Figure 3.3** it can be seen the data has been classified with a circle function, which have been done through transforming the 2D data into 3D data such that it can be linearly separable with a single plane. After being split is again transformed to the original 2D so it can be seen a circle.

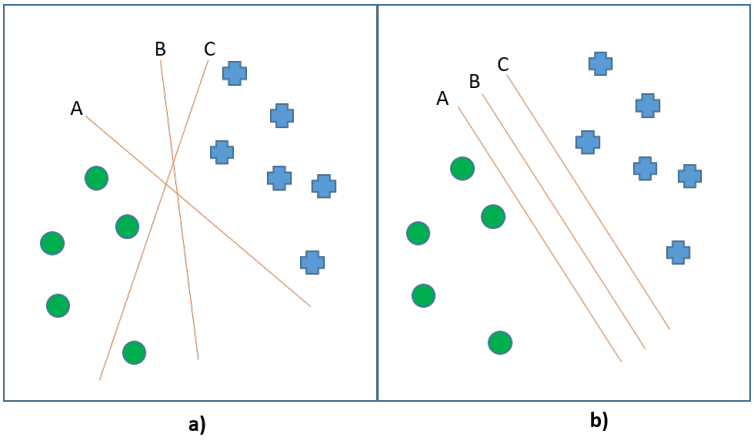

Figure 3.1: Classification Problem: a)Identifying the right hyperplane type 1, b) Identifying the right hyperplane type 2

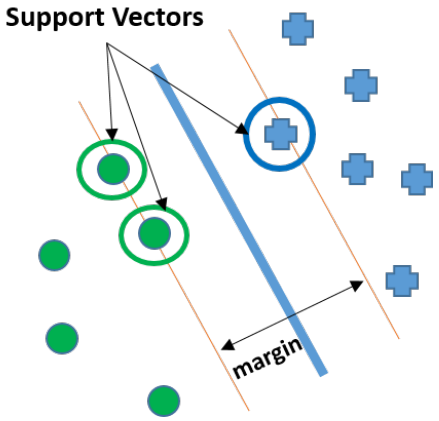

Figure 3.2: SVM illustration

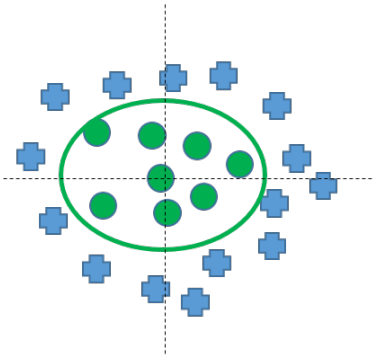

Figure 3.3: SVM kernel trick

3.1.2 Ensemble Model

The idea of Ensemble models is that combines multiple 'individual' models together to deliver superior prediction power. Ensemble is a supervised learning technique that combines multiple *weak learners* algorithms to get better predictive performance that could be obtained from any of the learning algorithms by itself. Among the different ensemble models it is found:

- **bagging**: This ensemble method create a random subsets from the training data and then builds a classifier for each subset. Finally the result of those multiple classifiers are obtained using average or majority voting.

- **Boosting**: This method provides sequential of the predictors. Trains the first predictor on the whole data set while the rest are learnt on the training set only based on the performance of the previous one. If classes are missclassified with the first learner, then it gives higher weight to the others classified observations. Iteratively, it continues adding classifier learners until getting a good performance.

- **Stacking**: Stacking method uses different classifiers to predict a class and then a new learner is used to combine their predictions with the aim of reducing the generalisation error.

By using these methods the benefits are better prediction and more stable model and it is also less noisy than other models. Hence, due to the difficulty of the amber gemstone database, this method is considered to classify them.

3.2 Deep Learning (Deep Neural Network)

The concept of Deep Learning comes as a different technique for implementing machine learning. The method comes from the neural networks (NN) technique in machine learning which works a bit like a brain by tuning itself through lots and

lots of training to learn one specific class. Unlike a biological brain where any network can connect to any other neuron within a certain physical distance. These artificial neural networks have discrete layers, connections, and directions of data propagation. Each neuron assign a weighting to its input according to how correct or incorrect it is relative to the task being performed (**see Figure 3.4**). The final output is then determined by the total of those weightings. Those weights are tuned as much as it is got a good yield by analysing the accuracy in a different data.

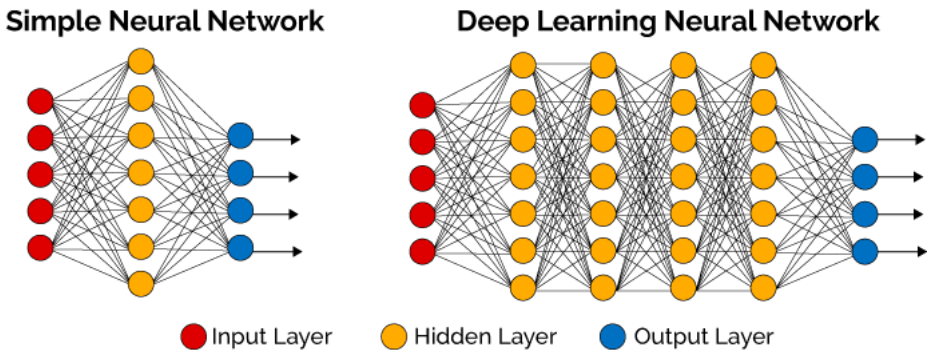

Figure 3.4: Comparison Simple Neural Network vs Deep Neural Network [2]

As a difference from Machine learning, in the deep learning concept it is taken a neural network with a bunch of layers and neurons that can run massive amounts of data like millions and millions of images from YouTube to be able to recognise the kind of music related to the user's taste or millions of images to classify a thousand of different objects.

In summary, Deep learning is a class of machine learning algorithm that:

- Use a cascade of multiple layers of nonlinear processing units where each layer uses the output from the previous layer as input
- Learn in supervised (classification) and/or unsupervised (pattern analysis). Both methods are used during this project.
- Learn multiple levels of representations that correspond to different levels of

abstraction [1].

Modern deep learning models are based on artificial neural networks, whose variation in deep learning are called deep neural network (DNN). Among the types of DNN it can be found the convolutional deep neural network (CNNs), recurrent neural networks (RNNs), Faster R-CNN based on region proposal network. However, a lately technique that now is being used more and more for real time object detection is the named YOLO (You Only Looks Once), another kind of deep neural network which combines all the concepts from CNN,RNN and Faster R-CNN. Here in this report it will be presented only Convolutional deep neural network and YOLO, since they are the methods that have been used to carry out the researching.

3.2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (**CNN or ConvNet**) is a type of deep neural network that use a variation of multilayer perceptron designed to require minimal pre-processing. As the ordinary Neural Networks they are made up of neurons that have learnable weights and biases in which each neuron receives some inputs, performs a dot product and optionally is followed by a non-linearity. They also have a loss function on the last (Fully Connected layer) layer which it is tried to minimise by applying different techniques of optimization. The difference then with the ordinary networks is that it is assumed that the inputs are images, allowing in that way to encode certain properties into the architecture to make the machine more efficient and reduce the amount of parameters in the network. CNNs use relatively little pre-processing compared to other image classification algorithms, which means no previous feature extraction to the images is carried out.

Regarding to the Architecture of the network this consists of an input and an output layer as well as multiple hidden layers. The hidden layers consists of convolutional

[1]Process of extracting attributes from an object or system

layers[2], pooling layers and fully connected layers (**see Figure 3.5**).

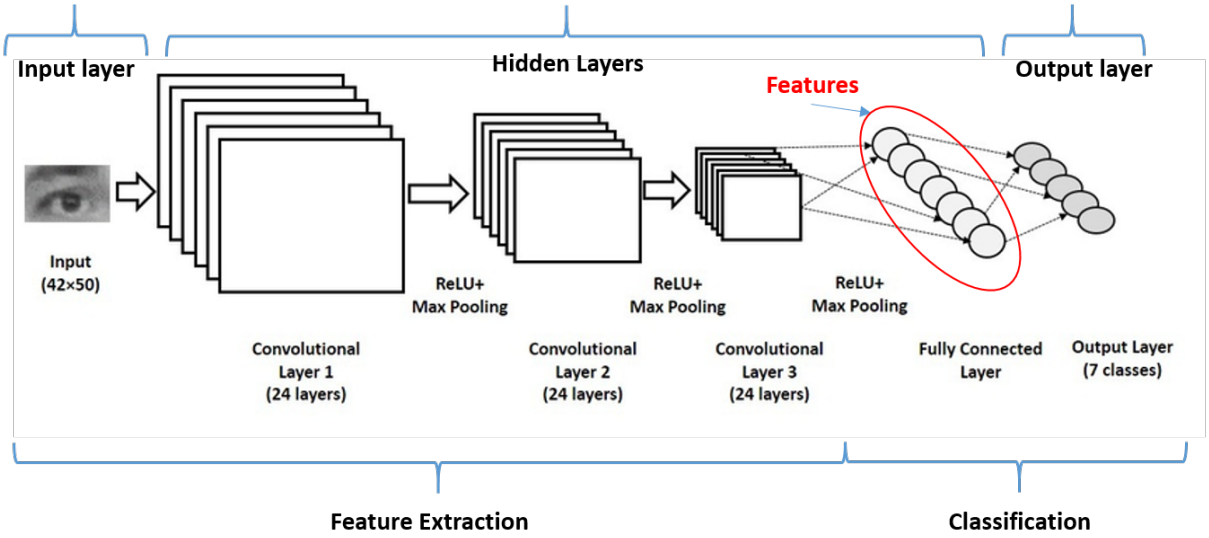

Figure 3.5: CNN-Convolutional Neural Network architecture

- **Input layer**: In the input layer it is normally checked the size of the image in order to be consistent with the structure of the architecture, since each architecture is made for an specific image size.

- **Convolutional layer**: In this layer it is applied a convolution operation to the input with the filter, passing the result to the next layer in such a way that in every layer it is extracted new features from the input image. Thus, the more convolutional layers the more features it will be obtained; however that does not mean the better performance it will be got. It is important to know that the *filter* is referred to as a neuron or kernel and that the array of numbers in the filter are called *weights*. Those filters are thought of as **feature identifier** (edges,colors,curves,etc). **Figure 3.6** shows how the filter acts as feature extractor and in **Figure 3.7** it is appreciated how the filter (feature) is convolved with some part of the image. When it matchs the feature with

[2]The convolution layers is rather cross-correlation layers than convolution, since mathematically it is performed the cross-correlation between the weights (filter) and the input (image)

the image the output becomes stronger and when not, the convolution output is cero.

- **RELU layer**: After each layer and before the pool layer it is convention to apply a non linear layer (activation layer) like the REctified Linear Unit layer which is merely an element wise operation (applied per pixel) that replaces all negative pixels values in the feature map by zero:

$$f(x) = max(0, x), \text{Where x is the value of a pixel} \tag{3.1}$$

- **POOL layer**: The pooling layer after the RELU layer it is used for a down-sampling operation along the spatial dimensions (width,height). It is mean reduces the dimensionality of each feature map but retains the most important information. They can be of different types: Max, Average, Sum, etc. It is mostly applied *Max Pooling* which takes the largest element from the rectified feature map within a window (normally 2x2) with an specific stride [3]. **Figure 3.8** represent an example of max Pooling done with a filter 2x2 with stride = 2. In the top part of the picture it is seen how the max pooling is performed and in the bottom part it is shown how strong it can be reduced a features of size 224x224x64 to 112x112x64.

- **FC (i.e. fully-connected) layer**: This layer has a full connections to all activations in the previous layer ("Fully Conneted") that works as a tradictional Multi Layer Perceptron that uses a softmax activation function in the output layer. Bear in mind that also other classifiers like SVM can be used instead, since these layers perform the function of a normal classifier using the high-level features from the convolutional and pooling layers. The softmax function is used to reduce the output of each unit between [0-1] and divides them such that the sum is 1. Thus, it is obtained a probability to belong

[3]Stride is the number of sliding steps when performing the convolution

to one class per each output. **Figure 3.9** attempts to explain the output of the softmax activation where it can be seen that for each class it is given a probability which determines to which class belongs each input.

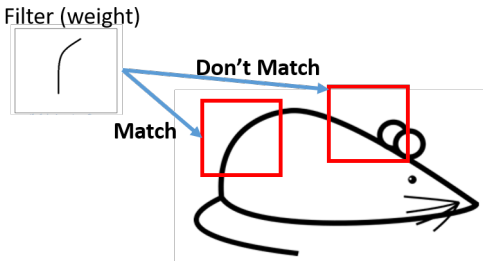

Figure 3.6: Representation of a filter of the CNN trying to look for a feature in the image

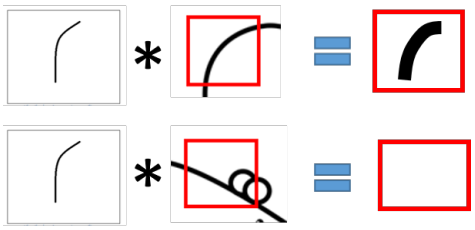

Figure 3.7: Convolution of a filter with some parts of the same sample in Figure 3.6. At the top, result of the filter matching some shape of the mouse. At the bottom, result of the filter not matching the specific shape of the mouse.

It is important to take into account some basic concepts regarding to the design of any convolutional neural network; they are the *padding*, *stride* and the *size of the filter*. In the one hand, the **padding** is used in any convolution operation in order to recover the original size of the image. So it is the number of zeros it is added at the borders needed to not lose the size. This depends directly of the **size of the filter** which is the size of the window used for the convolution operation. Thereby, the formula for knowing the size of the padding is:

$$Padding = \frac{K-1}{2}; \text{K: filter size} \tag{3.2}$$

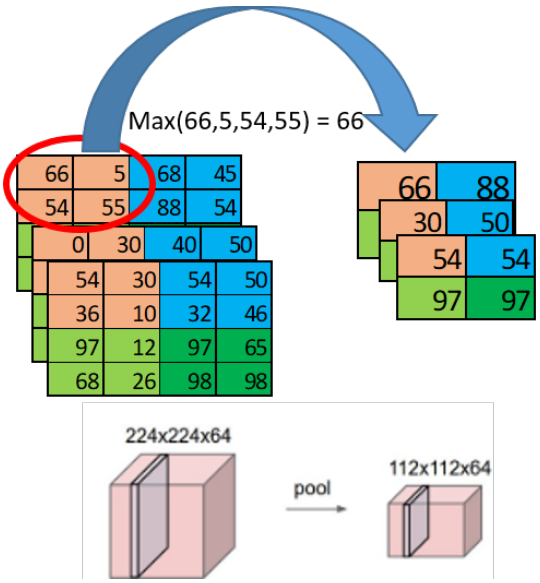

Figure 3.8: Representation of the Max Pooling operation to reduce dimensionality of the features.

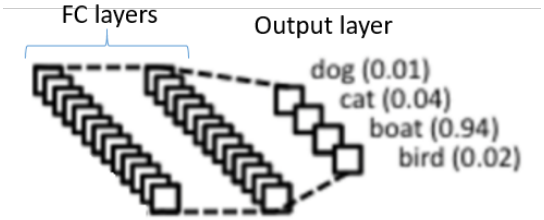

Figure 3.9: Fully Connected layers with the softmax activation that give some probability for each class

On the other hand, the **stride** is the number of sliding steps for the window to perform the convolution. By changing the stride, it can be changed drastically the dimensions of the image. In summary, in order to know which will be the size of the output after every convolutional layer it is used the next formula in function of the stride, padding and size of the filter:

$$O = \frac{W - K + 2 \cdot P}{S} + 1 \tag{3.3}$$

O: output height/length; W: input height/length; K: filter size; P: Padding; S: Stride

3.2.2 YOLO

You only look once (YOLO) is the state-of-the-art of the real time object detection system. This is extremely fast and accurate processing images in real-time at 45 frames using the basic YOLOV2 and compared to the state-of-the-art detection systems, YOLO makes more localisation errors but is less likely to predict false positives on background [6]. The difference of YOLO from other object detector is that uses a single CNN network for both classification and localisation using bounding boxes. As a matter of comparison of YOLO with other object detectors **Figure 3.10** shows the mAP (mean average precision) of the latest version YOLO V3 against the rest of detectors.

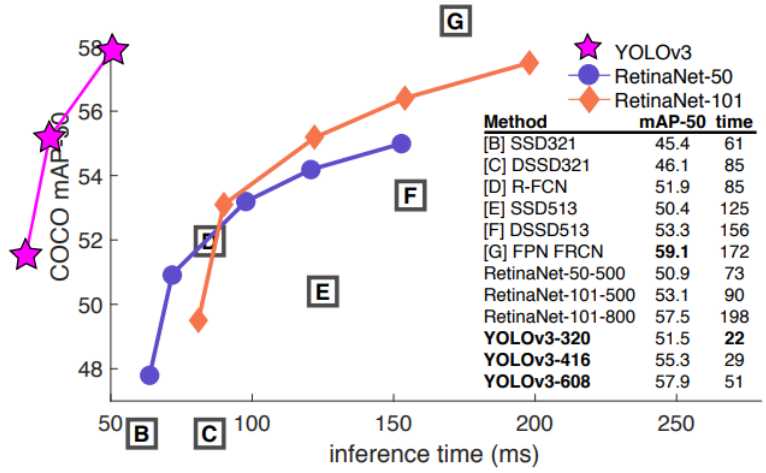

Figure 3.10: Comparison of the mAP of different Real Time Object Detection techniques(**source:** [4])

The thing is, any classifier like VGGNet or Inception can be taken and turned it into an object detector by sliding a small window across the whole image. At each step is ran the classifier to get a prediction of what kind of class is inside the current window and at the end take the ones that are the most certain about. Obviosuly, as it is performed the classifier many times the approach is going to be too slow. YOLO works with a different approach, instead of sliding window and region proposal-based techniques, this sees the entire image during training and test time,

in such a way that implicitly encodes contextual information about classes as well as their appearance only once.

The system divides the input into an S X S grid (normally 13x13). If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object taking into account *non-max suppresion* which is discarding all boxes with low probability ($P_r(Object) \leq 0.6$) and any remaining box with IoU [4]≥ 0.5. Each bounding box is responsible of detecting 5 predictions: $x, y, w, handC$. The (x, y) are the center of the box relative to the bounds of the grid cell, (w, h) the width and height, respectively, and C is the conditional class probabilities, $Pr(Class_i|Object)$. Finally, at test time it is multiplied the conditional class probabilities and the individual box confidence predictions to get the final results (Eq 3.4). **Figure 3.11** explains the model in a graphically way, starting from dividing the grid cells until getting the final detections.

$$Pr(Class_i|Object) \cdot Pr(Object) \cdot IoU = Pr(Classi) \cdot IoU \tag{3.4}$$

By doing this, it is obtained the final confidence score which indicates the probability that a bounding box contains a specific type of class. These scores combines the probability of a specific class appearing in a box and how well the predicted box fits the object.

The final result of YOLO per each grid cell is a tensor with the information of the position of the box $(, y, w, h)$, the probability of a box containing an object $(P_r(Object))$ and the class probabilities. Therefore, for a number of predicting bounding boxes B, a number of C class probabilities and a split into SxS grid cells of the image, leads to a tensor of dimensions S x S x $(B \cdot 5 + C)$.

[4]IoU is Intersection Over Union and indicates the relation between Area of overlap and the Area of Union of the predicted and real object

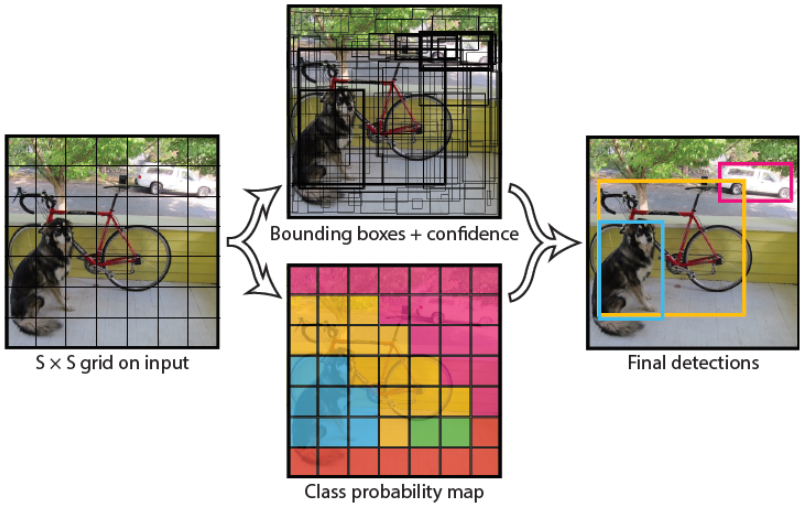

Figure 3.11: The YOLO model. It divides the image into an *SxS* grid, for each grid cell predicts *B* bounding boxes, confidence for those boxes, and *C* class probabilities. **source : [6]**

Regarding to the CNN arquitecture that YOlO uses, **Table 3.1** shows the latest structure YOLO V3, whose architecture is named **darknet-53** because of its 53 convolutional layers.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 3.1: Arquitecture of YoloV3 (**source: [5]**)

3.3 Database: Amber gemstone

Amber gemstone database have been extracted from a current project in development. This consist in classifying different amber stones which are passing through a conveyor belt and with a camera in front the conveyor to be able to get the data from and then process them to classify them as required. **Figure 3.12** shows an sketch of how the system have been built to get the database for the presented work.

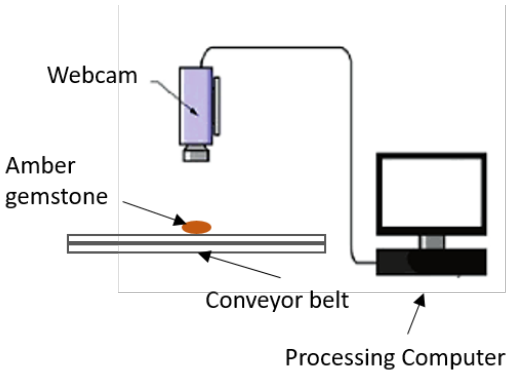

Figure 3.12: Illustration of the system made for extracting the data and classify them

After having been extracted the data, they have been classified one per one extremely carefully by experts. Classify is not an easy task, since most of them are affected by the brightness so it is normally needed a view from different point of views in order to properly asses to which type of amber belongs each one.

Figure 3.13 shows the group of 12 classes that have been chosen for classify them. Those samples have been tagged by experts according to colour into 4 main groups:

- **Clear-Semiclear** (Class01-Class06): Transparent with a yellow hue and some acceptable opacity.
- **Opaque** (Class07) : Opaque with small transparent areas.
- **Crystallised** (Class08-10): Mainly opaque with many bright-coloured spots.

- **Darkish**(Class11-12): Black or dark coloured with small bright areas.

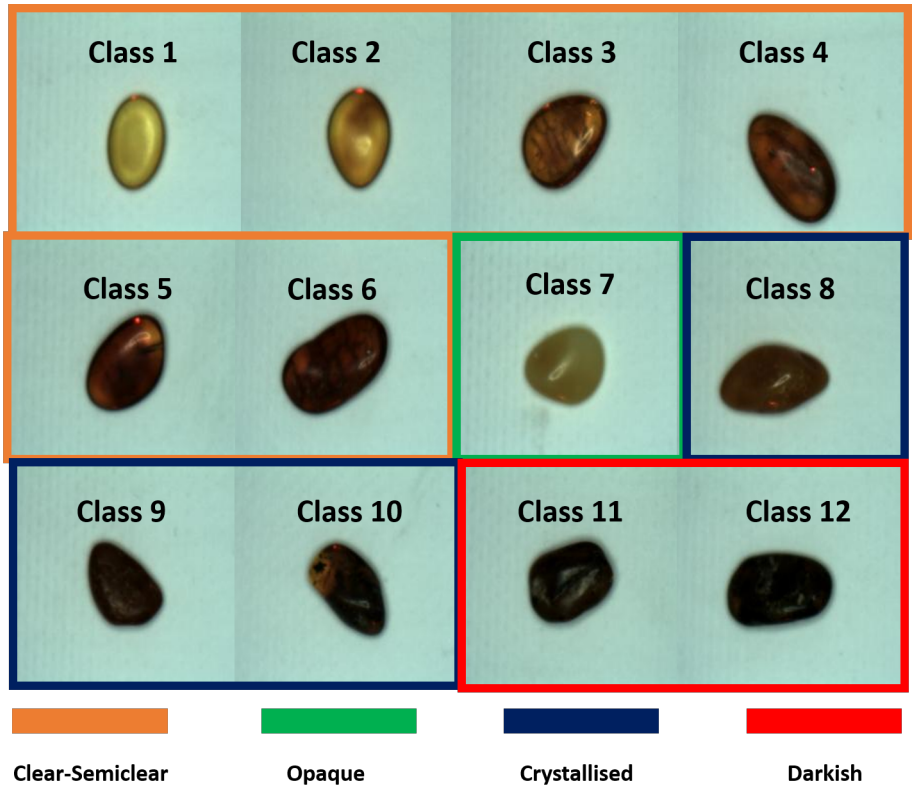

Figure 3.13: Amber gemstone data with 12 classes and grouped by their characteristics

The original pictures from the database have an specific size of 640x480 which in turns has a meaning of 307200 different features per image where most of them are background. After analysing a fair amount of images it has been detected that the information of amber stone in each one are ranged between 9% (the smallest) and 13%(the biggest), which means only that small percentage of features are usefull for the machine learning.

The challenge in the classification of amber stones is that some gemstones are more translucent than others which make them strongly sensitive to the light. Thus, nearby classes can be easily miss classified when they are affected by the light environment. **Figure 3.14** shows the first group of nearby classes which showcases the

problem of the light. It is seen how difficult is to recognise them for being classified. **Figure 3.16**, shows the last group of four nearby classes. Although, class 10 looks to be really affected by the brightness, it can be appreciated that the others are not as affected by the light as the first group of gemstones.

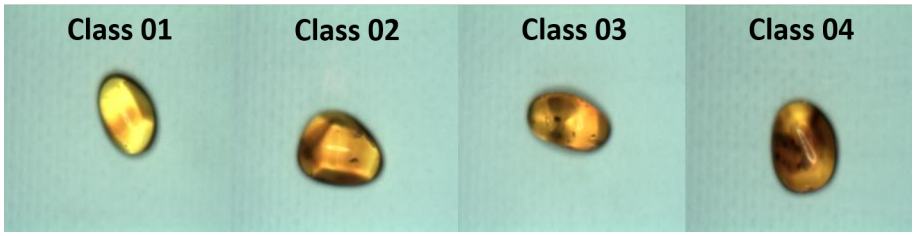

Figure 3.14: Amber gemstone data with 12 classes

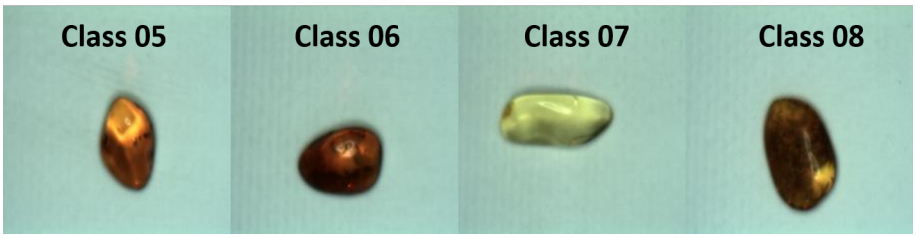

Figure 3.15: Amber gemstone data with 12 classes

Figure 3.16: Amber gemstone data with 12 classes

3.3.1 Modifications of the original database

The original size of the image may need to be resized depending on the different convolutional neural network structures used or may need to be preprocessed before in order to achieve different performances. So that, two types of modifications have been thought from the original database to get different performances. **Cropping**

the image to an specific size instead of resizing it and **cropping an small part from the center** to have a purely colour sample. All those modification wil be carried out by using the code in **Annex B.4**.

Database type 2: Cropping the image to an specific size

As a matter of researching of a good performance it have been thought this new modification of the original database. As it may be well known, every time it is used a new architecture of a specific CNN it is needed for the image to be resized. Resizing an image means resampling from the view of the Nyquist sampling theorem. Theoretically, it should not be a problem since after resizing it is got a good quality image; however, as what it matters is to keep as much features of the color of the amber as possible, this method may not be the appropriate one.

It was also observed that the background of the amber are taking too much pixels up and they are useless pixels for the machine learning. Hence, the approach of this idea is to try to focus only in the amber stone by cropping it to an specific size according to the CNN architecture used in order to keep as much features as possible.

Figure 3.17, illustrates this approach by comparing resizing and cropping. The example shows a final result of image size 227 x 227 to be able to fit in the AlexNet architecture, which will be the one used during this thesis. **Figure 3.18** shows some results of the original database cropped to 227x227. It can be observed that for some stones this size is not enough, so some part of the shape are missed. However, what is important is the color features rather than the shape.

Unlike the original database which contains only between 9% and 13% of amber stone, in this modification it is able to have between 48% (the smallest) up to 72.2% (the biggest) of information of the amber features per image. So it means that the machine learning can learn more small features of the texture of each amber

gemstone.

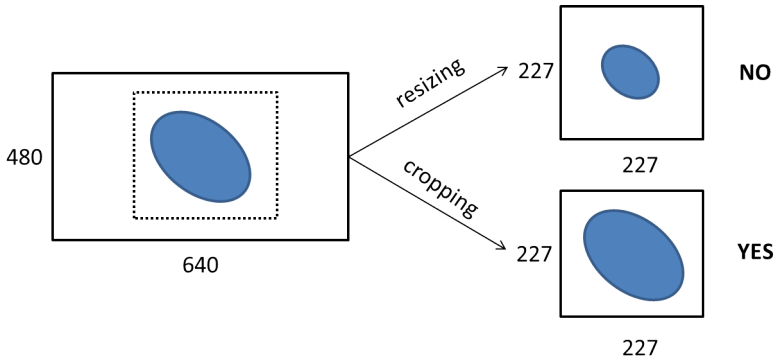

Figure 3.17: Difference between resizing and cropping the original image for getting the database type 2. Final result 227x227 image for AlexNet architecture

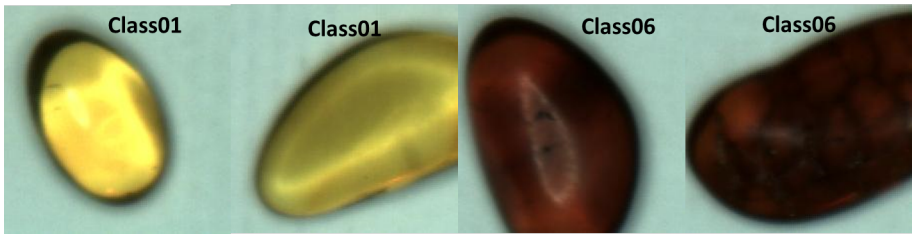

Figure 3.18: **Data Set 2:** Original images cropped into 227x227 for AlexNet architecture instead of resizing them

Database type 3: Cropping an small part from the center

This new modification is presented as a last modification of the original database to get different performances in the accuracy. The idea comes from the desire of getting purely colour features rather than any shape of the amber stone, since as already mentioned this is not too important when classifying them. The procedure is the following:

- It is detected the center of the amber stone.
- It is cropped from an small square which contains only the colours, not background at all. In this case a size of 81 x 81 have been found as appropriate.

- It is resized from 81 x 81 to 227 x 227 according to the AlexNet architecture.

The procedure is explained in **Figure 3.19** and in **Figure 3.20** it can be seen some results of the image processing carried out to get the new database type 3. It can be appreciated the color and texture feature of the image.

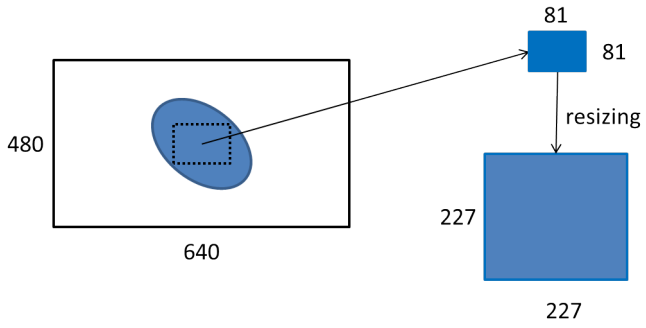

Figure 3.19: Scheme of the image processing carried out to get the database type 3

Figure 3.20: **Data Set 3:** Original images cropped into 81x81 from the center of the amber stone

Chapter 4

Methods applied

In this chapter it will be carried out the explanation of all the methods that have been used during the work. Due to this thesis is about researching the method of deep learning to improve the accuracy of the state of the art of classification of amber stones, it is tried the three most common strategies: **Transfer learning**, **Training from scratch**, and **feature extractor + non linear classifier**. Finally, a new strategy it is tested aroused from the transfer learning technique that will also be presented later on.

It is not clear which method will produce a better performance or which will work better in training set, validation set or test set, since during the feature extraction, looking at the colours will be the main priority other than the shape. Learning the shape as a feature for classifying amber stones can lead to miss classification since there are multiple shapes for each amber class and similar shapes among them (**see Figure 4.1**). So that, during the training the filters can learn some shapes in the first layers of the CNN which consequently could be a problem for a good classification. However, the training features must be focused on colour layers, so the shapes may be irrelevant.

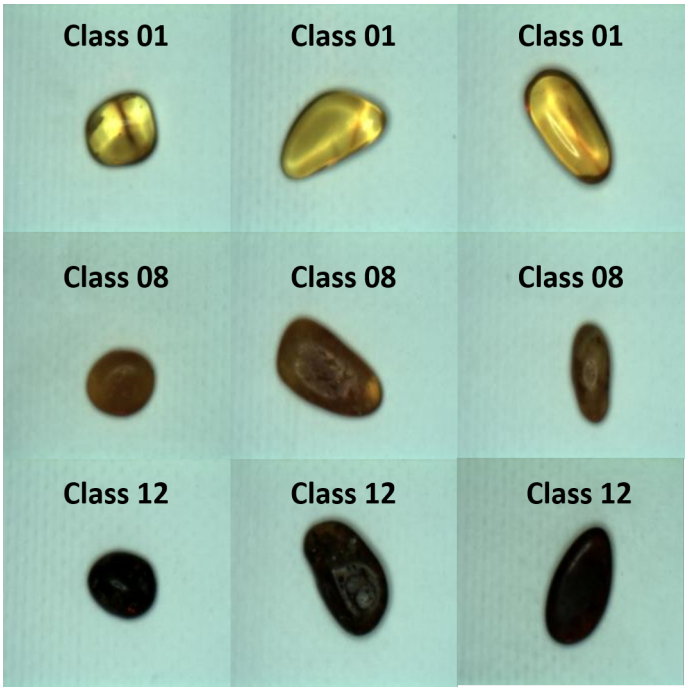

Figure 4.1: Shapes of the amber stones

4.1 Transfer Learning

Transfer learning technique it is the first technique in which it have been thought since it is a rapid progress or improved performance when modeling a second task with new data set using deep neural networks learning. Transfer learning is about using knowledge from a model that has learned when solving one classification problem using a specific data set, in a new task with a different dataset by "fine-tunning" the same model. Instead of starting the learning process from scratch, what this technique does is starting from weights and parameters that have already been learnt in a trained model.

This is motivated by human learning since normally people can transfer their knowledge learnt to other situations. for instance, some mathematics can use the maths for computer science or some professional table tennis player can transfer their skills to tennis. Humans can learn in many domains and can also transfer from one do-

main to other domains. Hence, applying the same idea to deep learning models, one model that have already been trained using a large amount of image database posses weights and filters that can share to other models to classify new items just by training the last layer (Full Conv). **Figure 4.2** illustrates how transfer learning works. In such example, the filters and weights (features) that have been used for classification of cats,cars,dogs and airplanes, are used for classify new items like bicycles, people, birds and balloons.

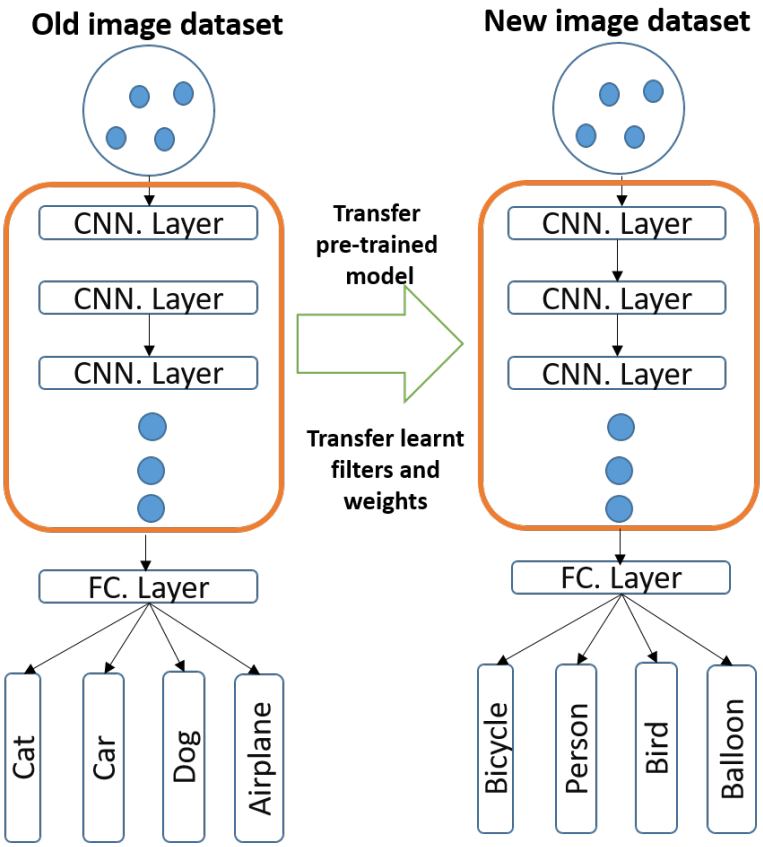

Figure 4.2: Transfer Learning scheme

As it can be seen from the picture above the old model is the pre-trained model for the new model which will act as a feature extractor. Thereby, it is needed only to train the classification part, which is the last layer (Fully Connected layer). Nevertheless, the weights of the other layers work as a starting point of the training. The fully connected layer is removed and replaced it with the specified classifier. Next it

is presented a summary of the steps to follow for constructing a pre-trained model approach

- **Select Model**. There are many models like Alexnet, Google Inception Model, Oxfor VGG Model or Microsoft ResNet Model. So the first step is to choose one of those models as a pre-trained model.
- **Reuse Model**. The model chosen can then be used as the starting point for second task of interest. This involves using the same architecture and the same weights learned.
- **Tune Model**. Besides using the same architecture of the chosen model, optionally, the model may need to be adapted or refined in their parameters. Moreover, new images may need to be adapted to the input layer.

This method usually used when there is no enough labelled training data to train the network from scratch and when there already exists a network that is pre-trained on a similar task which is usually trained on massive amounts of data. The advantage is clearly seen because as it is not training from scratch and the data used does not come in huge amounts, it is saved a lot of time.

Regarding to the different techniques proved taking the knowledge of the transfer learning, two extra methods have been thought. On the one hand, it is tried to perform data augmentation only on training data and let the validation and test data with the original data (**Method 1**). On the second hand, it is tried to augment the training and validation data letting the test data with no modification (**Method 2**).

Bear in mind that the training of the transfer learning technique is carried out in MATLAB, other than the data pre-processing. For all the following methods a different variations will be tried regarding the image processing:

- **Mod-0**: Data augmentation with some disturbances in the image like gauss filtering, sharpening, rotation and rotation + noise (salt and pepper).
- **Mod-1**: Data augmentation with no noise and disturbances at all. Only rotations of the original image.
- **Mod-2**: Transfer learning using the weights of the highest accuracy obtained during the assessment of the learned weights on test data along the epochs.

4.1.1 Method 1: Data augmentation on training data

Taking advantage of the data augmentation technique for obtaining more data due to the lack of them, in this method it is tried to carry out an augmentation of the data used for training which will be a 60% of the original data. Then, the rest of the data will be split in 10% for validation and 30% for testing. In **Annex A.1** and **Annex A.2** can be found the code for splitting and training the data.

This method is performed only with the original database with no modification. Next it is explained the augmentation that have been done in the training set (**Mod-0**):

- **Gauss filtering**: Blur the image thorugh a gauss filter of $\sigma = 2$.
- **Sharpening**: Enhance the image by applying a high pass filter (see [11] for more details).
- **Rotation**: Rotation of the image -90 degrees.
- **Rotation + noise**: Add noise(salt and pepper type) and rotate -90 degrees.

Figure 4.3 shows some example of the final results after the augmentation performed. The total number of images is 2218 so from that the training set is 1332 (60%), the validation set is 222 (10%) and the testing set 664 (30%). With the

augmentation performed the number of training data increases up to 6660 images. **Figure 4.4** shows some trials of this method to have a first idea of how the training evolves during the epochs. It can be observed that the accuracy of the training data is around 90% whilst the accuracy in the validation is 75.91%. So it differs too much one from the other since this the validation and training data have different pre-processing. However in chapter 5, more trials will be presented and assesed.

Figure 4.3: Data augmentation carried out for the **Method 1-Mod-0**

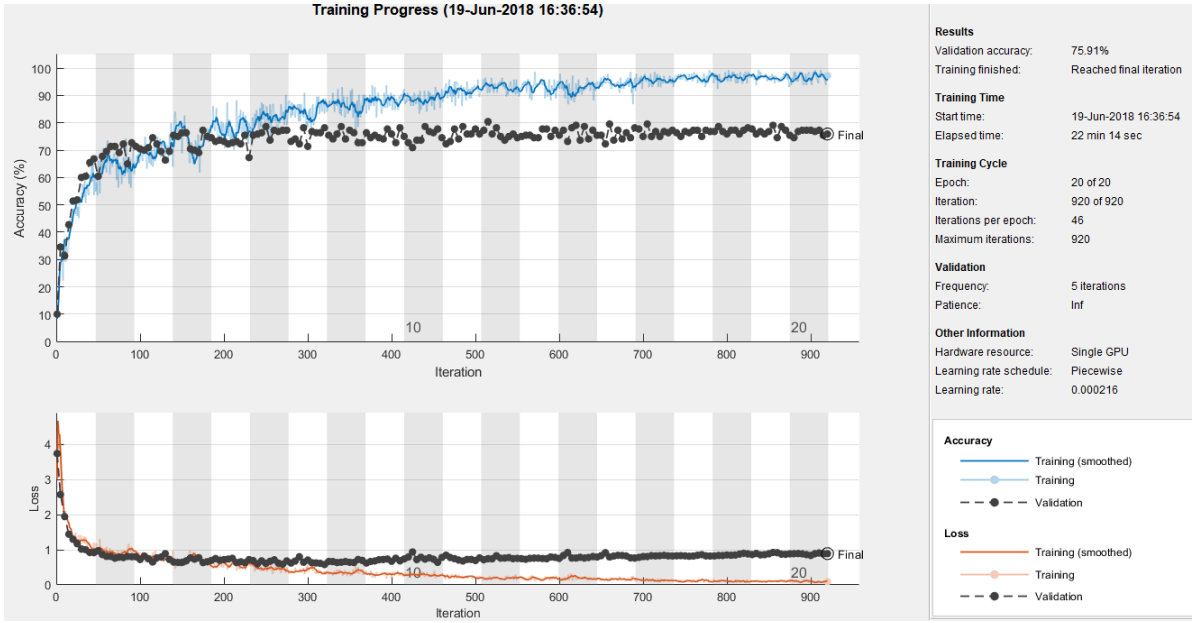

Figure 4.4: Example of result of some training with the **Method 1-Mod-0**. Note that this is not the best accuracy

4.1.2 Method 2: Data augmentation on training and validation data

The second strategy that will be carried out in this work is augmenting not only the training set but the validation set so it is possible to evaluate better with the same augmentation. After, it is evaluated in the test set which remains with the original data. As a variation of this strategy, two more methods have been tried with the different modifications of the database. The main purpose is to try to prove whether a different pre-processing of the images can lead to better accuracies or not. For this method, a code for splitting the data and for training it is slightly changed regarding to method 1. They are found in **Annex B.1** for splitting and **Annex B.2** for training.

In this method there will be tested also the three variations explained in **chapter 4.1**. As a matter of demonstration here it will be presented how the modification type 2 have been done, since modification type 1 and type 2 are only variations in the data augmentation. First, it is saved all the weights from each epoch during the training. Later it is used those weights to asses how is the evolution in test data when using such weights (**see Figure 4.5**). This is done with the code found in **Annex B.3**. After evaluating, it is picked the weight of the epoch that shows the highest accuracy to finally use it as starting point(**see Figure 4.6**) for the next training.

Method 2.1: Data augmentation using Database type 2

This technique uses the database type 2 (**see chapter 3.3.1**). With this modification where is not resized but cropped and where is obtained as much as possible only the features of the amber removing the background, is expected to get one of the best performances. However, this will depends only in the CNN architecture and the data augmentation.

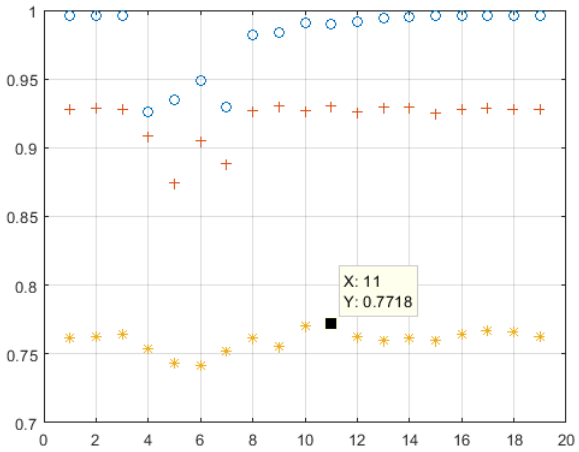

Figure 4.5: Example of the performance evolution along the epochs in Testing Data. From here the maximum accuracy is taken, which is in epoch 11

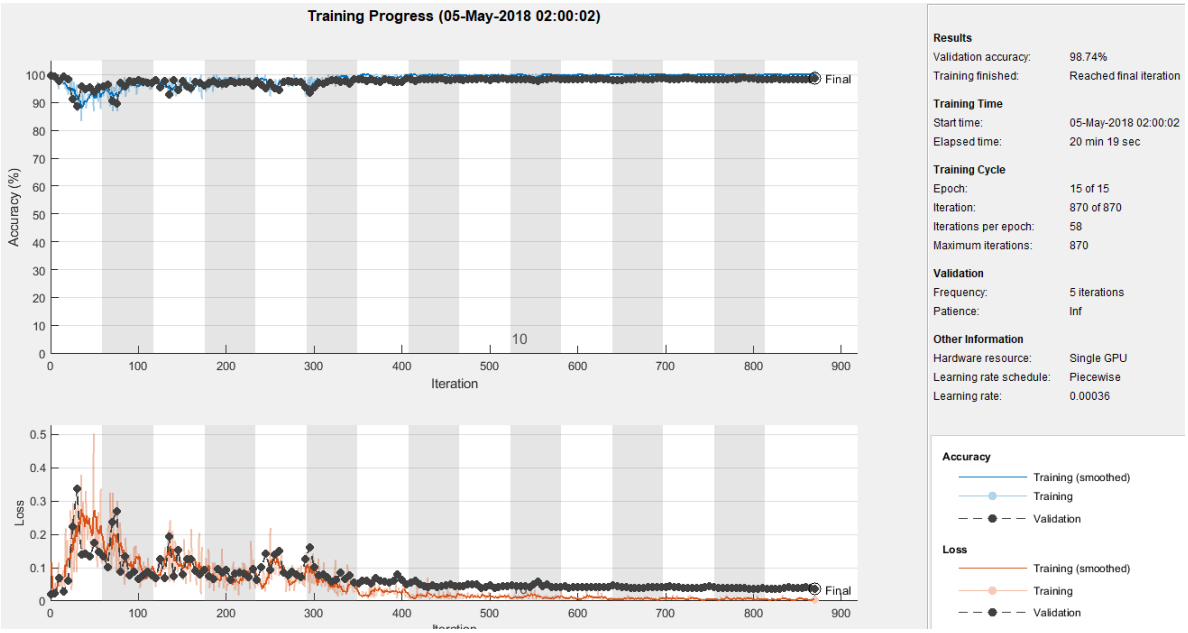

Figure 4.6: Example of transfer learning using as starting point the weights of the epoch that give the highest accuracy in Test data

The same strategy of modifications (Mod-0, Mod-1, Mod-2) as in the method 2 and method 1 will be applied to evaluate different accuracies. **Figure 4.7** and **Figure 4.8** show an example of the data augmentation carried out for the **Method**

2.1-Mod-0 and for the **Method 2.1- Mod-1**. For the **Method 2.1-Mod-2** it will be chosen the augmented data that gives the best accuracy.

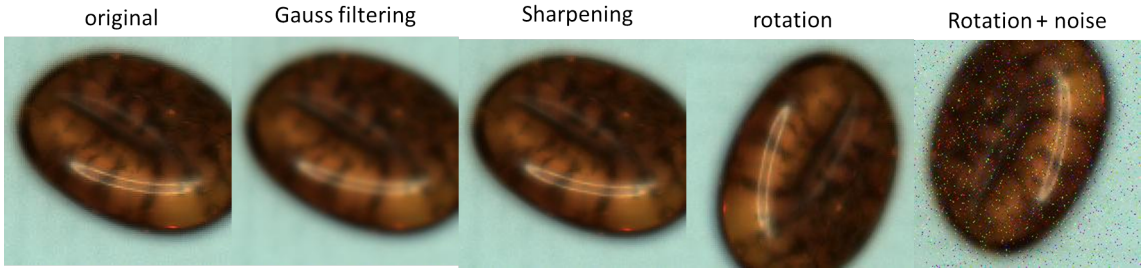

Figure 4.7: Data augmentation carried out for the **Method 2.1-Mod-0**

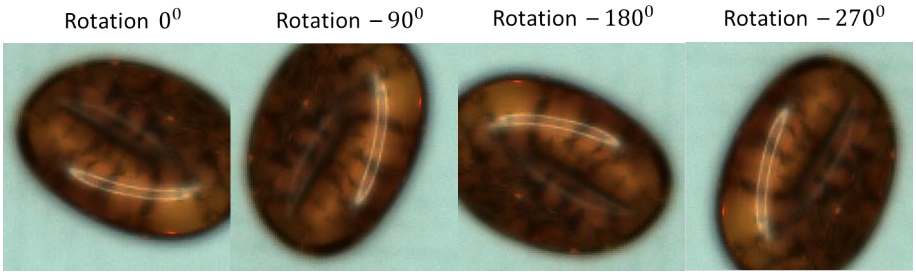

Figure 4.8: Data augmentation carried out for the **Method 2.1-Mod-1**

Method 2.2: Data augmentation using Database type 3

In this last strategy a new database is tried and is the database type 3 (see chapter 3.3.1), which is to crop from the center a small sample of the texture of the picture, with no background as disturbance. The idea here is then try to learn all the features of the color and texture of each class since the shape is irrelevant. The disadvantage with this method is that cutting an small sample[1] leads to resizing such sample to adapt it to the AlexNet architecture (227x227) or any other architecture, which means a bunch of fake augmented features for each image. Exactly if the cropped image is 81x81 so 6561 features and resized to 227x227, 51529 features, there will be (51529-6561) fake features that the machine learning will train. Nevertheless,

[1]The sample cropped is of size 81x81 in order to avoid background at all

this is not use to be a problem since when resizing it is tried to keep as much as possible the same characteristics of the image. **Figure 4.9** and **Figure 4.10** shows some results of the modification type 0 and type 1 in the data augmentation for this method.

Figure 4.9: Data augmentation carried out for the **Method 2.2-Mod-0**

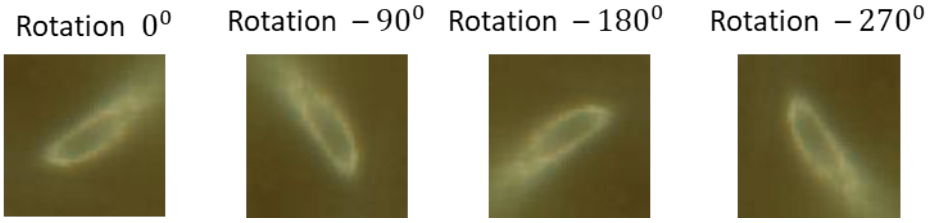

Figure 4.10: Data augmentation carried out for the **Method 2.2-Mod-1**

4.2 Training from scratch

After making tests and trainings by using transfer learning technique, the next method to try in deep learning is training the whole data from scratch. Training from scratch means using as starting point random weights (**see Figure 4.11**) and filters instead of having some weights that have been trained in a large amount of data like in transfer learning. It is mean that the loss will be quite big at first and then will decrease slowly after fairly enough epochs. It has been included this method as a part of the researching due to the architectures that have been used for transfer learning have been trained in data that are quite different from amber gemstone data. Lots of the data used for AlexNet are objects with different shapes rather than amber data which only the texture matters and that can vary according

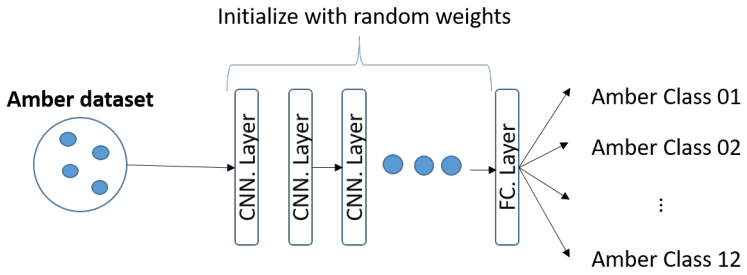

Figure 4.11: Scheme of training from scratch

to the brightness making even more difficult the task.

It will be used YOLO V3 (You Only Looks Once) in order to perform the training from scratch. As already explained in **section 3.3.2**, YOLO is the best real time object detector achieving good accuracies in videos with higher frames per second. Nevertheless, the goal in this thesis is classification in images that only contains amber stones so, the task of localising it may result useless. Therefore this method may not work as good in classification as in localization problems.

4.2.1 Method 1: Localization Database type 1

This method is about using YOLO V3 for its main purpose which is localization and classification at the same time. In order to perform the training there is some previous steps and modifications to follow to adapt the CNN architecture. The source for the darknet code and pre-trained weights and configuration files are found in **[1]**. Next is presented all the previous steps to start training and the commands to train and predict.

- **Step 1**: Create a yolov3-amber.cfg file with the same content as in *yolov3.cfg*. This is the configuration file where the CNN architecture is specified. In this file change the number of classes per $class = 12$ in the lines 610, 696, 783 and the number of filters en convolutional layers (lines 603, 689, 776). The number of filters is given by the formula $3 \cdot (nClasses + 5)$.

- **Step 2**: Create a file **obj.names**[2] with the names of the classes (eg.: cat, dog, car,...).

- **Step 3**: Create a file **obj.data** which must contain the information of where is located the training data, the test data, the class labels and to specify the backup to save the weights of the training (**Figure 4.12**). The text file to generate the training data information and test data information is found in **Annex C**.

```
classes= 12
train  = data/trainImages.txt
valid  = data/validImages.txt
names = data/amber.names
backup = backup/
```

Figure 4.12: Example of the obj.data information

- **Step 4**: Change the names of the images by "file/*class name*" and put the folder in the directory build/darknet/x64/data/obj/ .

- **Step 5**: Create a .txt file for each .jpg image file in the same directory and with the same name where it must be specified the object number (starting by 0) and the object coordinates in the image. For each object must be sorted like <object-class> <x> <y> <width> <height>. Bear in mind those coordinates are relative to the image and that (x,y) are the center not the corners (**see Figure 4.13**)

- **Step 6**: Create a file train.txt and validation.txt with the filenames of the images specifying the class of the image as mentioned in step 4.

- **Step 7**: Change the anchor boxes in the lines 612,697, 782 by the anchors obtained from the set of amber images. To get that it is used the code: darknet.exe detector calc_anchors data/amber.data -num of_clusters 12 -width 640 -height 480.

[2]It is normally created in the folder build/darknet/64 data/. In any case can be saved wherever as long as is specified when running darknet

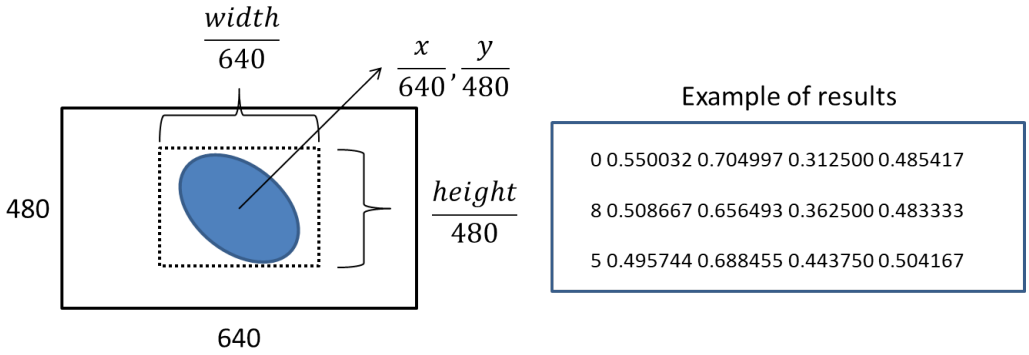

Figure 4.13: Example of how the object coordinate have to be specified (left) and some examples of the resulting .txt output (right)

- **Step 8**: Start training by using the command line: **darknet.exe detector train data/amberData.data yolo-amber.cfg**. The weights are saved in backup files with the name like yolo-amber_2000.weights, where 2000 is the number of iterations.

- **Step 9**: To predict once it is trained it is needed the code: darknet.exe detector map data/amberData.data yolo-amber.cfg backup/yolo-amber/2000.weights. With this code it gets the total accuracy and the accuracy per class.

4.3 Feature extractor + non linear classifier

The last strategy to test as a part of the researching of the best accuracy in classification of amber gemstones is mixing the deep learning and machine learning technique. As it has been studied the convolutional neural network method is divided into two parts: one part in charge of the feature extractor by using CNN layers, RELU layers and max pooling layers and the other part responsible of the classification of those features by using fully connected layers which behaves as a ordinary neural network. This approach tries to divide the CNN architecture by using only the layers for feature extraction in order to classify them using classics classifiers of machine learning like Super Vector Machine (SVM) or Ensemble (**see Figure 4.14**).

This method will be carried out by using transfer learning taking the architecture and the weights of AlexNet. Moreover, it will be tested the original database, the type 2 and type 3 database (see section 3.3) with the two different classifiers SVM and Ensemble. The code that performs this task is found in **Annex D**.

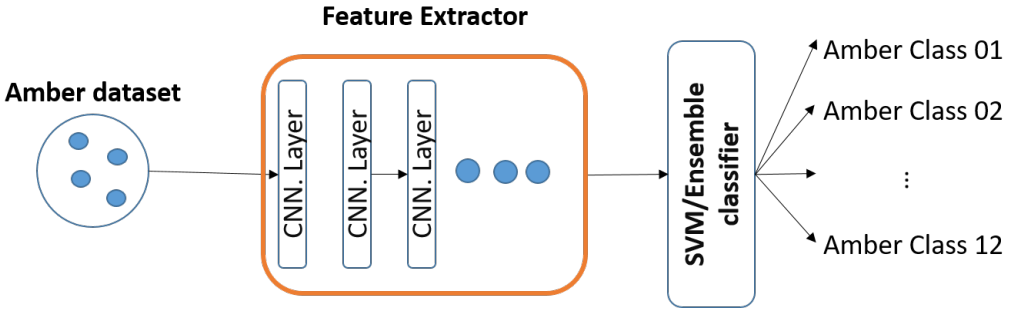

Figure 4.14: Scheme of the feature extractor using convolutional neural network and the classification task done by a non linear classifier as SVM or ensemble

In this case, the percentage for training data is of 70% and for testing data of 30%. There is no validation data. Data augmentation have been done only for the training set which leads to a 7765 different features and the rest for testing with no data augmentation leads to 666. **Figure 4.15** shows some results of this method by using SVM classifier and trained with the database type 3. It can be seen a pour accuracy and a lot of missclassification in class 10. However, in the next chapter will be presented all the results and proved such pour accuracy.

4.4 New transfer learning technique with double data augmentation

After trying the basic transfer learning, then the training from scratch and trying to take the features from the convolutional neural network trained and carry out the classification with some of the classical classifiers of the machine learning like

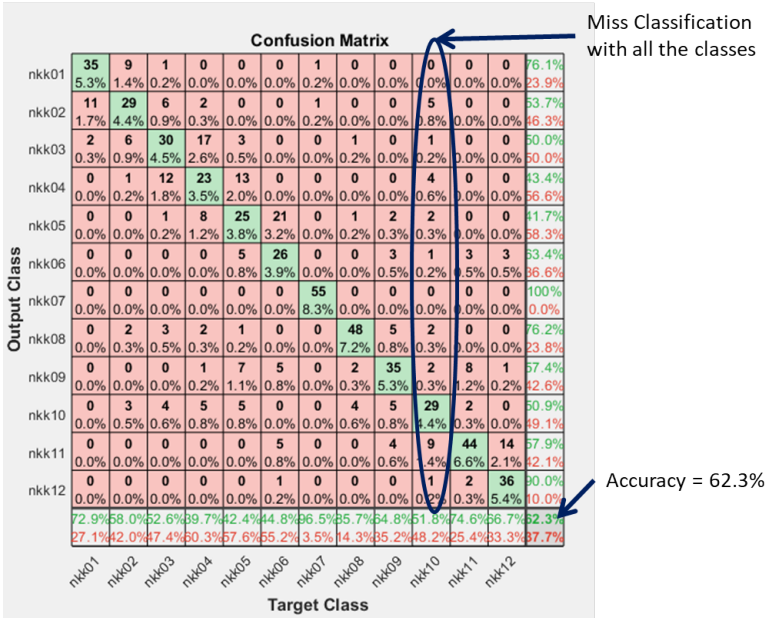

Figure 4.15: Example of the result of this method. This was tested with the database type 3 and with SVM classifier

SVM and Ensemble, it is tried a new approach. This new approach is based on the transfer learning technique which attempts to mix all three modifications explained in the **chapter 4 section 1** for a final accuracy. **Table 4.1** explain a short summary of each modification used for each method of the transfer learning technique.

MODIFICATION	DESCRIPTION
Mod-0	Data augmentation with noise and blurring
Mod-1	Data augmentation with no noise. Only rotations
Mod-2	Transfer learning starting from the highest accuracy epoch

Table 4.1: Summary of the modifications applied in transfer learning for each method

Figure 4.16 shows the scheme of this new approach clearly explained. The starting architecture for the transfer learning is the AlexNet with all their weights and layers trained in 1000 different objects. The data is divided into 60% for training, 10% for validation and 30% for testing. First a training is carried out with the data augmentation in training data with noise and blurring (**Mod-0**) and all the weights from each epoch are saved. After, it is evaluated the accuracy with the learned

weights all along the test data per each epoch and looked for the highest accuracy (**Mod-2**). Finally, by using such weights as starting point a second training is carried out with a different type of data augmentation in the training set; only rotations with no noise at all (**Mod-1**). The results for this method is presented in the next chapter.

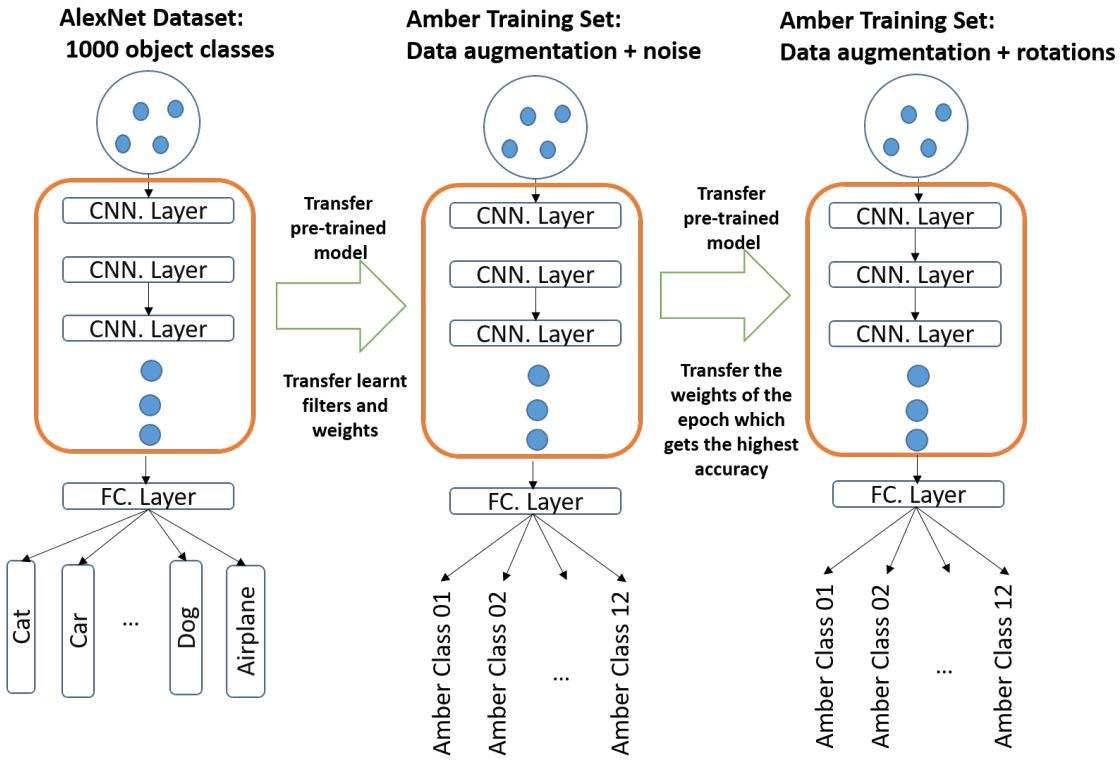

Figure 4.16: Escheme of the new approach using transfer learning technique

Chapter 5

Experimental Results

In this chapter it will be evaluated all the tests and trials that have been done along this thesis and that have been previously explained in the previous chapter. Here, it will be presented mostly more results about each method and it will be shortly explained separately the results per strategy: transfer learning, training from scratch, feature extractor + non linear classifier and the new transfer learning tecnique with double data augmentation. In the case of transfer learning strategy it will be showed the different methods with the different databases and different variations of data augmentation. Regarding to the databases, **Table 5.1** shows a summary of them with a short description.

	Description
DATABASE 0	Original Data with no pre-processing
DATABASE 1	Cropping the image to an specific size (227x227)
DATABASE 2	Cropping an small sample from the center (81x81)

Table 5.1: Summary of all the types of databases used for the researching

5.1 Results Transfer Learning

As a reminder of the many methods and trials that have been explained (**see chapter 4.1**). **Table 5.2** presents a short summary with all the modifications and variations of each method. Mostly all the methods rely on the databases and the type of data augmentation and processing.

	Data Augmentation	Description
Method 1 - Mod-0	Training Set	Data augmentation: Gauss filtering, Sharpening, rotation and rotation + noise using **DATABASE 0**
Method 1 - Mod-1	Training Set	Data augmentation: four rotations (0,-90,-180,-270) with no noise using **DATABASE 0**
Method 1 - Mod-2	Training Set	Transfer Learning using the weights of the highest accuracy evaluated on test data using **DATABASE 0**
Method 2 - Mod-0	Training - Validation Set	Data augmentation: Gauss filtering, Sharpening, rotation and rotation + noise using **DATABASE 0**
Method 2 - Mod-1	Training - Validation Set	Data augmentation: four rotations (0,-90,-180,-270) with no noise using **DATABASE 0**
Method 2 - Mod-2	Training - Validation Set	Transfer Learning using the weights of the highest accuracy evaluated on test data using **DATABASE 0**
Method 2.1 - Mod-0	Training - Validation Set	Data augmentation: Gauss filtering, Sharpening, rotation and rotation + noise using **DATABASE 1**
Method 2.1 - Mod-1	Training - Validation Set	Data augmentation: Gauss filtering, Sharpening, rotation and rotation + noise using **DATABASE 1**
Method 2.1 - Mod-2	Training - Validation Set	Data augmentation: four rotations (0,-90,-180,-270) with no noise using **DATABASE 1**
Method 2.2 - Mod-0	Training - Validation Set	Data augmentation: four rotations (0,-90,-180,-270) with no noise using **DATABASE 2**
Method 2.2 - Mod-1	Training - Validation Set	Data augmentation: four rotations (0,-90,-180,-270) with no noise using **DATABASE 2**
Method 2.2 - Mod-2	Training - Validation Set	Transfer Learning using the weights of the highest accuracy evaluated on test data using **DATABASE 2**

Table 5.2: Methods Applied by using the transfer learning technique

Before analysing the results of each method, some concepts have to be taken into account about the different tuned parameters that have been used during the training of the convolutional neural network. Those parameters are really important when training because can affect considerably the performance of the system. They are:

- **Learning Rate**: Learning rate is a hyper-parameter used to control the loss gradient when searching local minima. The lower the value, the slower descend

along the downward slope. Too large values can overshoot the minimum and it may fail to converge.

- **Epochs**: An epoch is a single pass through the full training set. If it is being used gradient descent for the optimization then the number of epochs is the same as the number of iterations. Otherwise, if it is being used stochastic gradient descent then the number of epochs depends on the size of the minimum batch and the number of samples.

- **Minibatch Size**: Instead of passing though all the training data one by one, it is chosen a minimum batch size (small amount of samples), so in that way a fixed amount of samples is taken for estimating the gradient in such a way it is not needed all the samples. Hence, if there are 2000 samples and the batch is 500 then only 4 iterations is carried out, which means is much faster training. Normally if the batch is too small it tends to reach high accuracies for the training, if it is too big it tends to leads to better accuracies for the validation but cause a dropping int the training accuracy

- **Learning rate drop factor**: This parameter drops the learning rate by a factor every few epochs. So the lower is the drop factor the lower is the learning rate. If the learning rate is too long then it may never find the local minima.

Next is presented the results of the best accuracies obtained in validation data and in test data from the different method using transfer learning. Accuracy in training data have been omitted because is not relevant for comparing results. From **Table 5.3** to **Table 5.5** it is shown the accuracies for the method 1[1]. It is seen that the accuracy is around 74% and 75% in test data, with any significant improvement in the three modifications of this method. The reason is due to the overfitting produced by the data augmentation. **Figure 5.1** is the training result of the best accuracy

[1] In this method is taking 60% of training set and augmented them, and divide 10% for validation and 30% for testing without any data augmentation, see chapter 4.1

obtained from this method. In that picture it is proved that there is a clearly overfitting in the training because of the big difference in the loss and so in the accuracy; almost 100% for training and 74% for validation.

	METHOD 1 - MOD - 0					
TEST	Learning Rate	Learning rate drop factor	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.0001	0.8	20	30	75.45%	72.40%
T2	0.001	0.6	20	114	76.81%	73.72%
T3	0.0001	0.6	25	144	78.18%	74.80%

Table 5.3: Best results of the method 1-mod-0

	METHOD 1 - MOD - 1					
TEST	Learning Rate	Learning rate drop factor	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	114	73.64%	74.47%
T2	0.001	0.6	20	144	73.64%	74.77%

Table 5.4: Best results of the method 1-mod-1

	METHOD 1 - MOD - 2					
TEST	Learning Rate	Learning rate drop factor	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	114	75.45%	74.17%
T2	0.001	0.6	20	114	75.45%	75.68%

Table 5.5: Best results of the method 1-mod-2

From **Table 5.6** to **Table 5.8** it is presented the results for the method 2 [2]. For this method, by using the different modifications it is barely seen some change in the accuracy too, always keeping around 77% as the best accuracy for test data. On the other hand, it is seen a big difference in accuracy of validation. In this case, the modification type 2 with 95.95% of accuracy seems to be overfitted regarding the accuracy in test data. The reason is because it is being used the original data

[2]In this method is taking 70% of training set (60%) and validation (10%) to perform data augmentation, and divide 30% for testing with the original data, see chapter 4.1

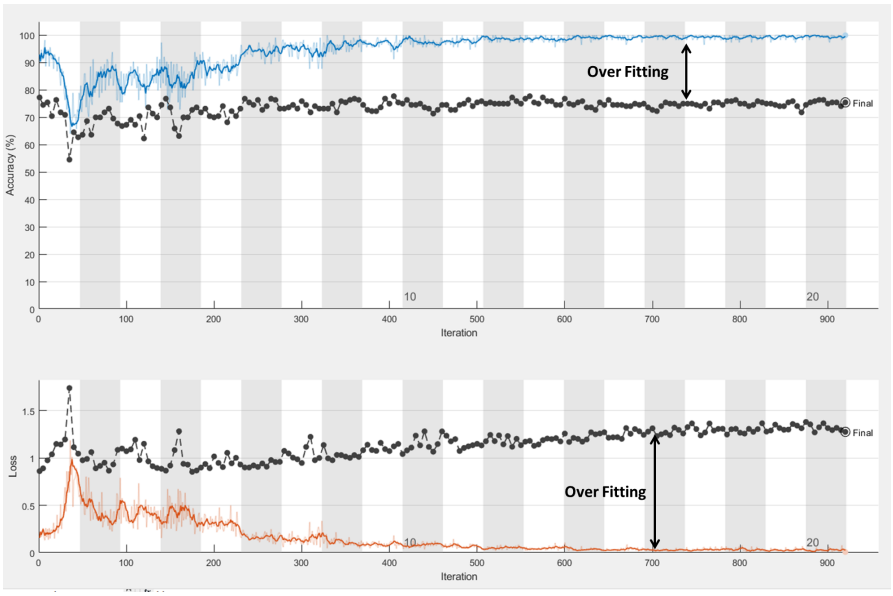

Figure 5.1: Demonstration of the overfitting taken from the best result obtained from method 1

for test and a modificated data with noise for validation which is the same as for the training. **Figure 5.2** demonstrate that as the type of the data is the same for training and validation there exist an small difference between them, so there is no overfitting; nevertheless when testing in different data this accuracy is dropping around 13% less.

	METHOD 2 - MOD - 0					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.0001	0.8	15	64	83.06%	74.40%
T2	0.001	0.6	15	104	88.38%	76.88%
T3	0.001	0.6	30	114	90.18%	77.18%

Table 5.6: Best results of the method 2-mod-0

Tables 5.9, 5.10 and 5.11 shows the results for the method 2.1 [3]. So far, with those accuracies it seems to be the best and that the best accuracy is found when in the data augmentation is applied only rotations, with no noise (method 2.1-mod-1). The reason here why this shows the best results is because of the database used.

[3]This method uses the database type 1, cropping the image to an specific size, see chapter 3.3.1

	METHOD 2 - MOD - 1					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	114	77.73%	74.77%
T2	0.001	0.6	20	244	79.98%	75.38%
T3	0.001	0.6	25	144	80.86%	77.02%

Table 5.7: Best results of the method 2-mod-1

	METHOD 2 - MOD - 2					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.2	20	114	92.79%	76.43%
T2	0.001	0.6	15	114	95.95%	77.03%
T3	0.001	0.6	20	128	99.37%	74.62%

Table 5.8: Best results of the method 2-mod-2

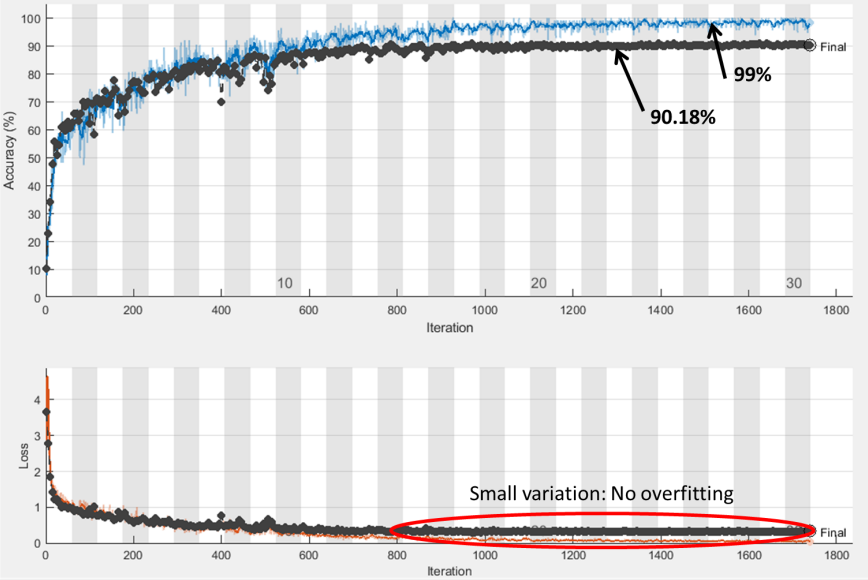

Figure 5.2: Demonstration of the best accuracy of the training in validation data due to the same type of samples.

The database base type 1, is focused on the amber stone with less background than the database type 0. In this database, it is found between 48% (the smallest stone) up to 72% (the biggest stone) of information of features rather than the pour percentage of features in the original database.

	METHOD 2.1 - MOD - 0					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	114	88.57%	78.23%
T2	0.001	0.6	20	144	87.23%	79.13%

Table 5.9: Best results of the method 2.1-mod-0

	METHOD 2.1 - MOD - 1					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	144	81.92%	79.80%
T2	0.001	0.6	20	104	82.92%	77.30%

Table 5.10: Best results of the method 2.1-mod-1

	METHOD 2.1 - MOD - 2					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	144	95.72%	78.83%
T2	0.001	0.5	20	144	89.70%	78.08%

Table 5.11: Best results of the method 2.1-mod-0

The last tables, shows the results for the last method 2.2 [4]. Surprisingly, this method shows the worst accuracies when was expected to get some better results due to the type of the database used. Such database only focuses in the colour and texture of the amber stone being 100% of only features. However, the best accuracy is obtained again by applying no noise in the data augmentation. One of the main problems here is that only an small part of the image is being used for the classification instead of the whole. So that, there is missing information which means that only one section of texture is not enough for classifying amber gemstones.

After seeing all the results separately per each method it can be concluded that the best accuracy for the strategy using transfer learning is for the **method 2.1-mod-1** with an accuracy of **97%** in training data, **81.92%** in validation data and

[4]This method uses the database type 3, cropping an small part from the center. See chapter 3.3.1

	METHOD 2.2 - MOD - 0					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.4	20	204	79.46%	70.22%
T2	0.001	0.6	20	104	84.23%	71.47%

Table 5.12: Best results of the method 2.2-mod-0

	METHOD 2.2 - MOD - 1					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	54	74.13%	72.67%
T2	0.001	0.6	20	244	73.68%	72.22%

Table 5.13: Best results of the method 2.2-mod-1

	METHOD 2.2 - MOD - 2					
TEST	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
T1	0.001	0.6	20	114	83.69%	71.02%
T2	0.001	0.6	20	144	84.30%	70.80%

Table 5.14: Best results of the method 2.2-mod-2

79.80% in test data. **Figure 5.3** shows the training stage of such method. It can be seen that from the epoch 10 it begins the overfit since at the end of the training there is almost 20% of difference between training and validation. Moreover the validation accuracy is not improving anymore from that point.

Figure 5.4 illustrates an analysis of the confusion matrix of the best results in test data. Here it can be appreaciated almost all classes have miss classifications not only in the nearby classes, which can be understandable but along the other classes too. Miss classification between the nearby classes can be acceptable since they are quite similar each other, but is not admitted to have many wrong detections with other not nearby classes. Class 09 and Class 10 belongs to the group of crystallised stones and due to the different bright-coloured spots they are mostly confused with class 04, 05 and 06 (clear-semiclear classes). On the other hand, the class with the best accuracy is the class 07, since is the only opaque class so it can-

not be confused with others. In any case, it can be said that the result for transfer learning technique is almost acceptable.

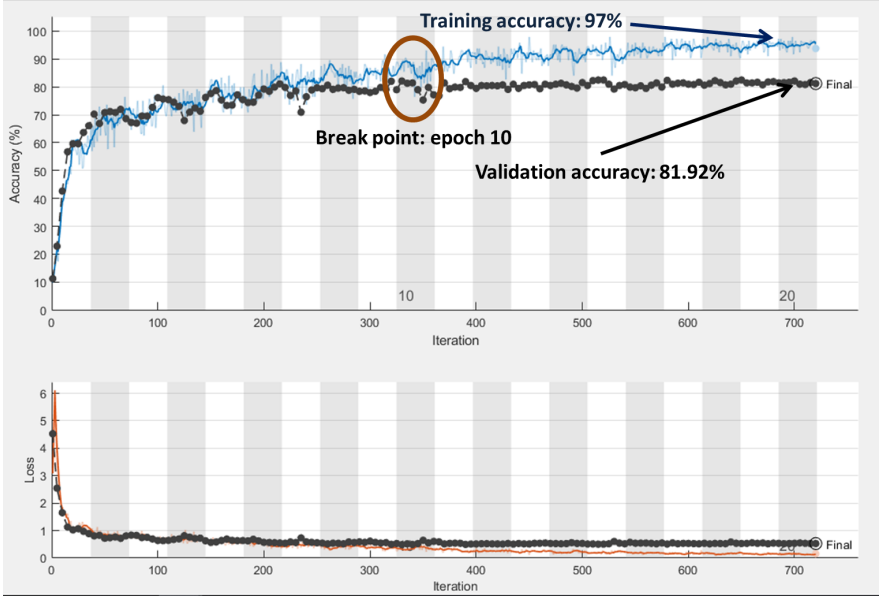

Figure 5.3: Analysis of confusion matrix of the best results in test data

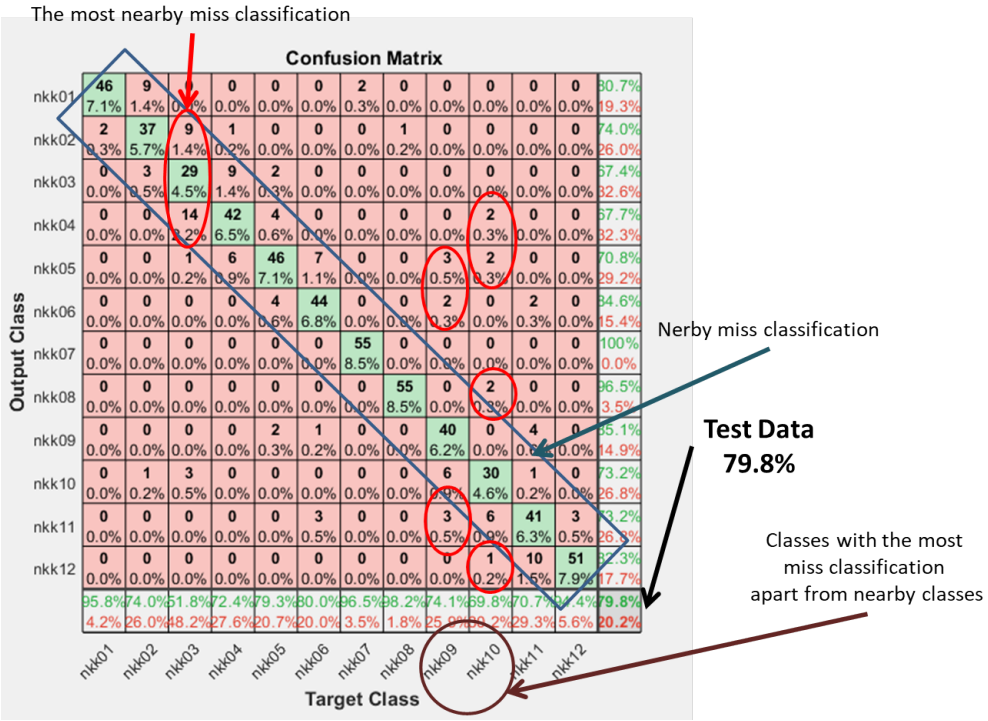

Figure 5.4: Analysis of the best results

5.2 Results Training from scratch

In this section it will be analysed the training from scratch and it will be proved how important is the quantity of samples taken for the training. As already explained in this case is used 70% for training and 30% for testing. There is no validation data and there is no type of pre-processing before for the images. **Figure 5.5** shows the procedure of the training stage. There, it can be seen an average loss of 0.386145 with a learning rate of 0.000001 and in the iteration 12017. In **Figure 5.6** it is showed the results of the training in test data analysed by its medium average precision (mAP). It can be seen a poor accuracy of 52.19%.

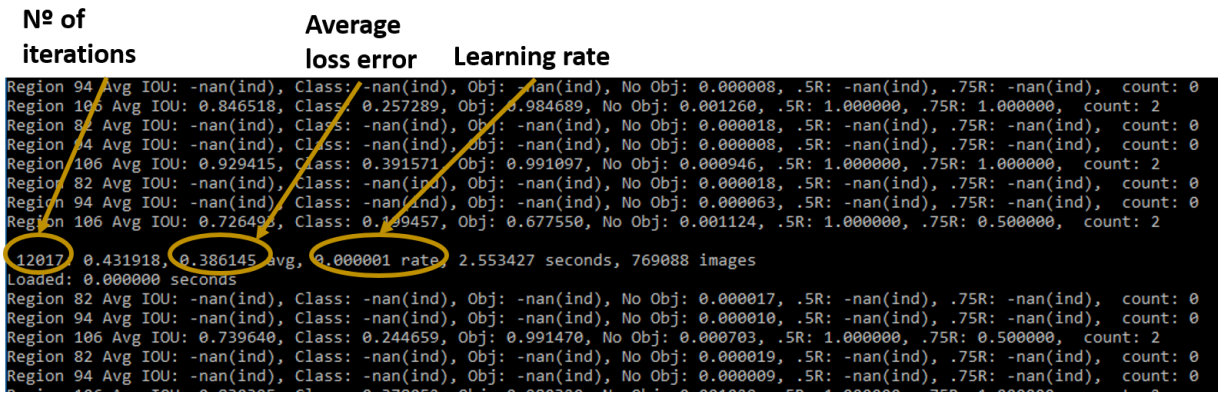

Figure 5.5: Training process from scratch

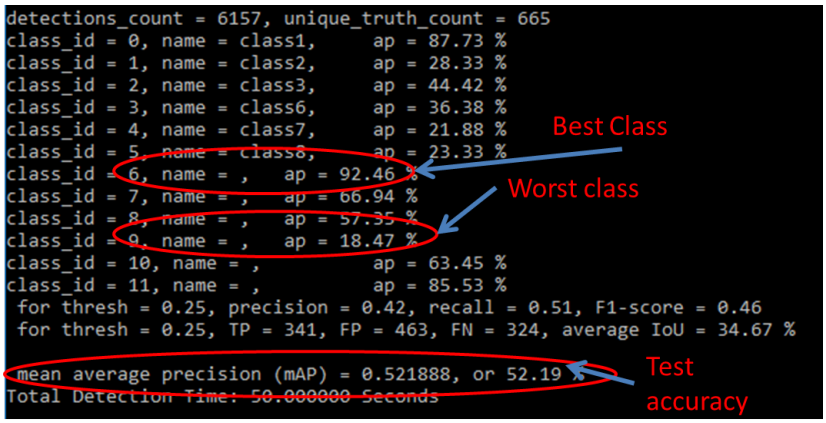

Figure 5.6: Results of the training from scratch after

The Figure above shows the best result after many iterations and it matches that the

class with the best accuracy is the class 07 again (opaque type) and the worst class is the 10 (crystallised type). Regarding to the time consuming, the training took around some days but the loss did not change too much since the 6000 iteration. **Figure 5.7** shows a part of the training loss from the 6000 iteration until 11000 iteration. With that image it can be concluded that the training from scratch is a hard task and with only 2218 total samples divided for train and validation is not enough. Normally, when training from scratch it is needed a huge amount of data so the neural network can learn better. In this case is around 200 samples per class, so it is not enough to learn all the complicated features of the amber gemstone.

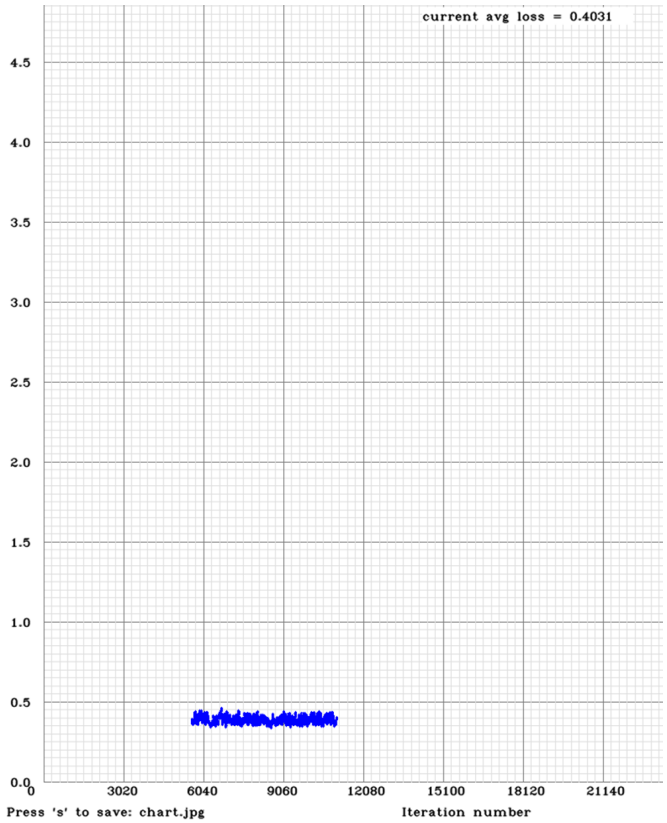

Figure 5.7: Average loss of the training from scratch

5.3 Results Feature extractor + non linear classifier

After transfer learning and learning from scratch, this section will present the results the strategy for featuring extractor with convolutional neural network + classification with a machine learning technique like SVM and Ensemble. Both classifiers are supposed to be the bests for non linear classification; reason why they have been chose as a matter of researching. Here, the three databases have been tested and all of them with data augmentation technique in training data performing only rotations. The data is divided in 70% training and 30% testing; there is no validation data.

The feature extraction is carried out by using the AlexNet architecture with all their weights and filters. Here, the task of SVM and ensemble classifier is to look for the best combination of features took from a 1000 classes (ImageNet) different from amber gemstones.**Table 5.15** shows the results for those tests.

With 7765 different features with a deepness of 4096 it is seen some pour accuracies in prediction on test data with the best of 71.02% using SVM classifier and using database type 1. In this case, works better with the database type 1 since it contains more information about the amber stone as already mentioned. So this helps to the classifiers to learn more features; nevertheless, none of those methods achieve the accuracy of the transfer learning technique.

CLASSIFIER	Database 0	Database 1	Database 2
SVM	67.57%	**71.02%**	62.31%
Ensemble	62.46%	64.40%	**65.47%**

Table 5.15: Results of the accuracies for each database and for each classifier SVM and Ensemble

5.4 Results of the new transfer learning technique

Finally, here is presented the last test tried; the new approach of transfer learning gathering different methodologies to reach only one simple accuracy as explained in **section 4.4**. This method has three steps in each of which is tried a different modification. The parameters used for each step is showed in **Table 5.16**. In that table is also seen the accuracy on test and validation data. For the first training it is obtained an accuracy of **74.17%** on test data and for the second training it is obtained **90.39%** of accuracy; a huge increment. This result is even bigger than the accuracy in validation of 89.55%. Recall that there is only data augmentation in training set and not in validation neither in test samples. So that, this could be one of the reason the validation accuracy is less than the test accuracy.

STEP	Learning Rate	Learning rate drop	Epochs	Minibatch Size	Accuracy Validation	Accuracy Test
1 (MOD-0)	0.001	0.6	20	114	75.45%	74.17%
2 (MOD-2)	BEST ACCURACY					**75.60%**
3 (MOD-1)	0.001	0.6	20	114	**89.55%**	**90.39%**

Table 5.16: Parameters and accuracy of each step of the technique

Figure 5.8 shows the confusion matrix of this method. It is demonstrated the accuracy of all the classes with the nearby classes have decreased considerably and what is more, the maximum amount of data per class miss classified different from the nearby classes is only one, with the exception of class 09 (crystallised type) which has 6 wrong detections.

It is also interesting to analyse the weights learned in each epoch during the training in the test data. That is explained with the **Figure 5.9**. There it has been tested the weights in the training set, in the validation set and in the test set. It can be appreciated the accuracy in training set reachs the 100% and what is even more interesting is that the accuracy in the 6th epoch in the test set reachs **93.69%**. The

main reason why it is got such higher accuracies with that methodology is because it is being trained twice with diferent data augmentation. In the first training with noise it learns some features that in the second training without noise does not learn. In that way it is created a powerful deep learning machine that can overcome the state of the art of the classification of amber gemstone.

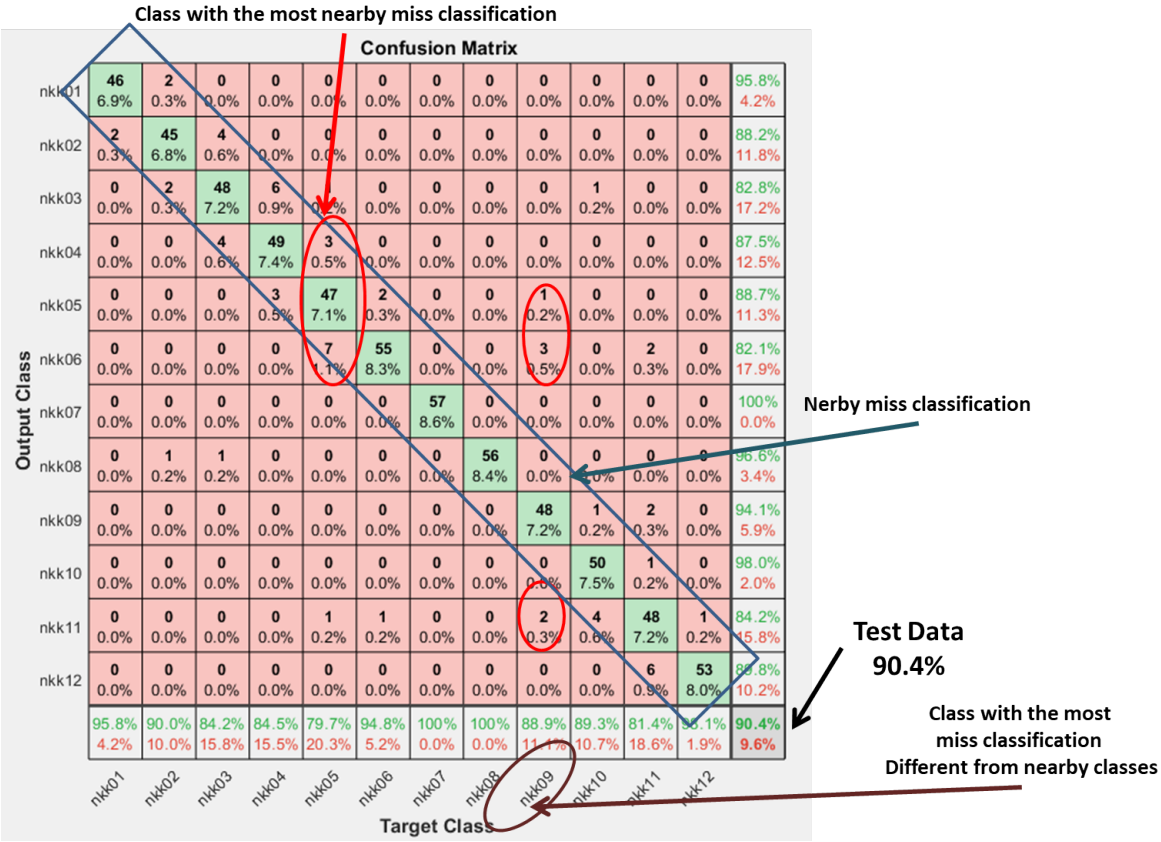

Figure 5.8: Best Confusion matrix ever

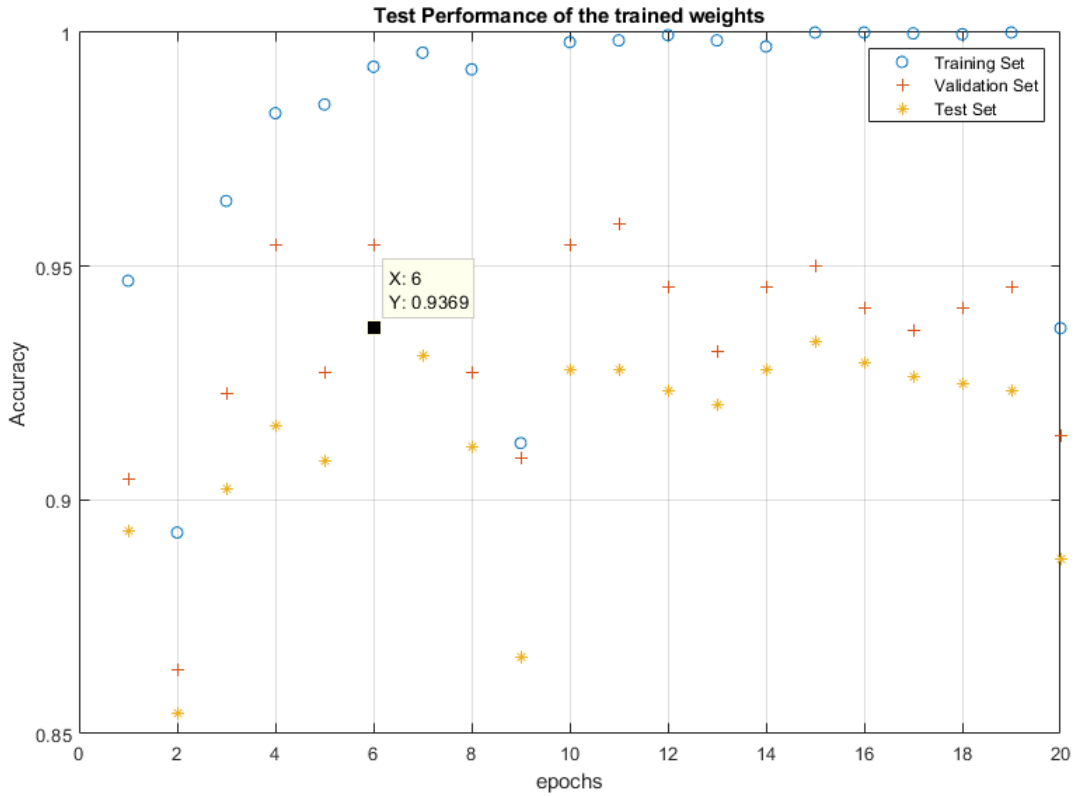

Figure 5.9: Best test performance

Chapter 6

Conclusion and future works

The researching of the different methodologies using deep learning to classify amber gemstone by its colors and texture it is hard task that mainly leads to disappointment when it is not possible to get good results in accuracy. During the work many trials have been tested in order to asses which method can give the good performance. The main problem that many researchers find when working with machine learning or deep learning is that it is never known whether the classifier it is being used will yield to good results. In neural network it is never known how many layers are needed to get a good classification with no overfitting or which kind of features will be better for the training such that it gets good accuracy in testing. This normally is too much time consuming and is even more time consuming to work with big data using deep learning. Training from scratch can takes days even working with GPU and not getting what is wanted in accuracy. In this work, many other ideas have been thought in training from scratch but due to the lack of time it has not been tested. Thereby, that is the reason why this presented thesis has mainly focused in transfer learning methodology.

Transfer learning has allowed to get the best results in classification. In the first method it was obtained 79.8% way much than the one got by training from scratch with 52% and than the use of Support Vector Machines as classifier with only 71.02%

of accuracy. AlexNet architecture trained in ImageNet database has been of a great utility, since is a network that have been trained in 1000 different classes; which means all the knowledge from those trained layers transferred to the training of the presented work has lead to a good accuracies for amber database.

On the other hand, the deep learning toolbox of MATLAB has allowed to manage with ease the work of deep learning and also MATLAB itself has been quite useful for image processing and for showing graphically the results obtained in training and in validation.

The definition of all the theoretical concepts previously given to the work, has been used to recall all what is known in machine learning and deep learning and also has been used to let the reader knowing which kind of concepts it is being used for carry out the researching. In the explanation of each method it has also been slightly introduced some theoretical part needed to understand each strategy.

Regarding to the performance of the classification it has been seen that the technique of transfer learning with double data augmentation has obtained the best accuracy with 90.4% of good classification. The reason is because it is being used transfer learning twice not with the same pictures but with different modifications. In the first training with amber images with noise the network learnt some features that helped way much to the second training with images with no noise at all but rotations to learn new features from amber and thus get the best accuracy. However, this is not the best accuracy, because during the training the accuracy is always fluctuating. After saving the weights per epoch and tested in test data it turns out that the best accuracy was obtained in epoch 6 is **93.69%**.

With such results it can be proved that this researching has overcome the accuracy of 88% of the latest work in amber stone classification. It also can be concluded

that deep learning technique can perform better than traditional machine learning techniques which requires a lot of effort trying to get the proper features for the classification. By using deep learning there is no need to preprocessing and extracting features because it does learn by itself. Therefore, it is only needed to research for the proper way of using deep learning in order to get the best results in any specific field.

As a future works, there is only left to keep doing more trials in the new modification of the databases (type 1 and type 2) with the last method that gets the best performance. It can also make some other types of data augmentation and to train three times instead of two. In training from scratch it can be tried to train with data augmentation and with the other databases too. As a last proposal, this work can be extended to classify more classes up to 20 or 30 by using transfer learning and new modifications different from what have been worked on this thesis.

Bibliography

[1] AlexeyAB. https://www.kdnuggets.com/2017/12/deep-learning-made-easy-deep-cognition.html, 15 de July de 2018.

[2] Favio Vazquez. `https://github.com/AlexeyAB/darknet`, 15 de July de 2018.

[3] Ioannis Valavanis, D. K. Multiclass defect detection and classification in weld radiographic images using geometric and texture features. *Elsevier* (2017).

[4] Joseph Chet Redmon. `https://pjreddie.com/darknet/yolo/`, 11 de July de 2018.

[5] Joseph Redmon, A. F. YOLOv3: An Incremental Improvement[on line]. *University of Washington, Allen Institute for AI* (2018.[Date of consultation: 12 of July of 2018]. Available in: `https://pjreddie.com/media/files/papers/YOLOv3.pdf`).

[6] Joseph Redmon, Santosh Divvala, R. G. A. F. You Only Look Once: Unified, Real-Time Object Detection[on line]. *University of Washington, Allen Institute for AI* (May 2016.[Date of consultation: 11 of June of 2018]. Available in: `https://arxiv.org/pdf/1506.02640.pdf`).

[7] Neelamma K. Patil, Virendra S. Malemath, R. M. Y. Color and texture based identification and classification of food grains using different color models

and haralick features, vol. 3 no 12. *International Journal on Computer Science and Engineering (IJCSE)* (Dec 2011).

[8] Saulius Sinkevicius, Arunas Lipnickas, K. R. Organic shapes classification by similarity to basic geometric shapes. *International Journal of Computer and Information Technology* (2014).

[9] Saulius Sinkevicius, Arunas Lipnickas, K. R. Amber gemstones sorting by colour. *ELEKTRONIKA IR ELEKTROTECHNIKA* (2017).

[10] Saulius Sinkevicius, Arunas Lipnickas, K. R. Automatic amber gemstones identification by color and shape visual properties. *Elsevier* (June 2014).

[11] Yao Wang. `http://eeweb.poly.edu/ỹao/EE3414/image_filtering.pdf`, 11 de July de 2018.

Appendix A

Codes Method 1

A.1 Code to split data: Training-Validation-Test

```
% DESCRIPTION: in this program we are going to create a set of
% training,validation and test data, 60%,10%,30%, respectively. In this
% case the data will be augmented is the trainind data and validation and
% test will remain the same. This code is for the method 1.
%%
clear all
close all
%% SPLIT TRAINING/VALIDATION/TEST
% Remove folder of training data if exist
if 7 == exist( 'trainingImages', 'dir')
  rmdir('trainingImages','s')
end
if 7 == exist( 'TrainValidImages', 'dir')
rmdir('TrainValidImages','s'); % remove the previous created before reading
    .
end
cd ..
allImages = imageDatastore('Training', 'IncludeSubfolders', true,...
```

```
    'LabelSource', 'foldernames');
cd Training
% Split Data 60% training, 10% validation and 30% test.
[trainImages, validImages, testImages] = splitEachLabel(allImages,
    0.6,0.1,0.3, 'randomize');
% save test and validation data:
if 1 == exist('TestValidData.mat','var')
    delete TestValidData.mat
else
    save TestValidData validImages testImages
end
%% Reading Images from training and validation to a post AUGMENTATION.
mkdir TrainingImages
TrainingData = readall(trainImages);
trainLabels = trainImages.Labels;

    dir=['TrainingImages'];
    eval(['cd ' pwd '\' dir]);
    prevLabel = char(trainLabels(1));
    l = 0;
    trainingLabels = [];
    for i=1:length(trainLabels)
        label  = char(trainLabels(i));
        % copy the same label per image according to the number
        % of data augmentation
        trainingLabels = [trainingLabels;
                          char(trainLabels(i));
                          char(trainLabels(i));
                          char(trainLabels(i));
                          char(trainLabels(i));
                          ];

        if label == prevLabel
            l = l+1;
        else l = 0;
```

```
52          end
53 %%%%%%%%%%%%%%%%%%%%%%  DATA AUGMENTATION WITH NOISE %%%%%%%%%%%%%%%%%%%%%%%%
54 % picture  = TrainValidData{i};
55 % imwrite(picture,[char(trainLabels(i)) '_' num2str(l) '.jpg'],'jpg');
56 % imwrite(imgaussfilt(picture, 2),[char(trainLabels(i)) '_' num2str(l) '_g.
       jpg'],'JPG');
57 % imwrite(imsharpen(picture),[char(trainLabels(i)) '_' num2str(l) '_s.jpg
       '],'JPG');
58 % imwrite(imrotate(picture,-90),[char(trainLabels(i)) '_' num2str(l) '_r.
       jpg'],'JPG');
59 % picture=imnoise(picture,'salt & pepper',0.03);
60 % imwrite(imrotate(picture,90),[char(trainLabels(i)) '_' num2str(l) '_d.jpg
       '],'JPG')
61
62 %%%%%%%%%%%%  DATA AUGMENTATION WITH NO NOISE (ONLY ROTATIONS) %%%%%%%%%%%%
63
64      picture  = TrainValidData{i};
65 %      picture=imresize(TrainValidData{i},[227, 227]);
66      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) '.jpg'],'jpg');
67      picture  = imrotate(picture,-90);
68      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 'f.jpg'],'jpg');
69      picture  = imrotate(picture,-90);
70      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 's.jpg'],'jpg');
71      picture  = imrotate(picture,-90);
72      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 't.jpg'],'jpg');
73      prevLabel = label;
74      end
75 cd ..
76 % save test labels:
77 trainingLabels = cellstr(trainingLabels);
78 save trainLabelData_type2 trainingLabels
```

A.2 Code to train data: Transfer Learning with Alexnet

```
% TITLE: Training CNN using AlexNet transfer Learning *METHOD 1*
% Description: This code train with the data augmented training set using
% AlexNet architecture and freezing only the last three layers.
clear all
close all
%%
load('trainLabelData_type2.mat');  % load the label saved from *split data*
all_TrainData= imageDatastore('TrainingImages');
all_TrainData.Labels = categorical(trainingLabels);
save TrainData all_TrainData    % save data for a posterior evaluation
%%
load('TestValidData.mat');% load test data an validation data

% STARTING ALEXNET
alex = alexnet;                          % Create alexnet network
inputSize = alex.Layers(1).InputSize;
layersTransfer = alex.Layers(1:end-3);  %layers transfer
numClasses = numel(categories(all_TrainData.Labels));% number of classes

% new layers to learn from 0
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'
        BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];

% Parameters to modify before training
opts = trainingOptions('sgdm', 'InitialLearnRate', 1e-3,...
```

```
    'LearnRateSchedule','piecewise',...
    'LearnRateDropFactor',0.6,...
    'LearnRateDropPeriod',5,...
    'ValidationData',validImages,...
    'ValidationFrequency',5,...
    'ValidationPatience',Inf,...
    'Plots','training-progress',...
    'MaxEpochs', 5, 'MiniBatchSize', 114,'Shuffle','every-epoch','Verbose'
        ,0,...
    'CheckpointPath','E:\Downloads\kevir_12_Alexnet\DataManagement\
        method1weightsT12');
%%%%% If needed to train from previous learnt network load the weights%%%%%
%  load('convnet_checkpoint__828__2018_06_22__00_48_31.mat'); %from T12.
 load('convnet_checkpoint__368__2018_06_22__02_52_58.mat'); %from T16

%  myNet = trainNetwork(all_TrainData, layers, opts); % train from alexnet
 myNet = trainNetwork(all_TrainData, net.Layers, opts);%train from previous
                                                       %networks.
%% TESTING AND BUILDING THE CONFUSION MATRIX.
% % Now let's the test the performance of our new "snack recognizer" on the
    test set.
Images = testImages;
Images.ReadFcn = @readFunctionTrain;    % Adapt images to 227x227
predictedLabels = classify(myNet, Images); % predict
accuracy = mean(predictedLabels == Images.Labels);% Compute accuracy
C = confusionmat(Images.Labels,predictedLabels);  % Build Conf. Matrix
```

Appendix B

Codes Method 2

B.1 Code to split data: Training-Validation-Test

```
1
2  %TITLE: Code to split Data in training,validation and test set (METHOD 2)
3  %DESCRIPTION: This code create a folder with the data used for training and
4  %validation. Also performs the data augmentation and save everything in
5  %.mat files. This is used for the method 2 with 70% training+validation and
6  %30% test data.
7  %%
8  close all
9  clear all
10
11 %% SPLIT TRAINING/VALIDATION/TEST
12 % Remove folder of training data if exist
13 if 7 == exist( 'trainingImages', 'dir')
14   rmdir('trainingImages','s')
15 end
16 if 7 == exist( 'TrainValidImages', 'dir')
17 rmdir('TrainValidImages','s'); % remove the previous created before reading
     .
```

```
18 end
19 %
20 cd ..
21 allImages = imageDatastore('Training', 'IncludeSubfolders', true,...
22     'LabelSource', 'foldernames');                    % Read from subfolders
23 cd Training
24 % Split Data 70% training + validation and 30% test.
25 [trainvalidImages, testImages] = splitEachLabel(allImages, 0.7, 'randomize'
       );
26
27 % save validation data:
28 if 1 == exist( 'TestData.mat','var')
29     delete TestData.mat
30 else
31     save TestData testImages
32 end
33 %% Reading Images from training and validation to a post AUGMENTATION.
34 mkdir TrainValidImages
35 TrainValidData = readall(trainvalidImages);      % Read images
36 trainLabels = trainvalidImages.Labels;           % copy labels.
37 %
38     dir=['TrainValidImages'];
39     eval(['cd ' pwd '\' dir]);
40     prevLabel = char(trainLabels(1));
41     l = 0;
42     trainingLabels = [];
43  % DATA AUGMENTATION:
44     for i=1:length(trainLabels)
45         label  = char(trainLabels(i));
46         % copy the same label per image according to the number
47         % of data augmentation
48         trainingLabels = [trainingLabels;
49                           char(trainLabels(i));
50                           char(trainLabels(i));
51                           char(trainLabels(i));
```

```
52                          char(trainLabels(i));
53                          ];
54
55          if label == prevLabel
56              l = l+1;
57          else
58              l = 0;
59          end
60  %%%%%%%%%%%%%%%%%%%%%%  DATA AUGMENTATION WITH NOISE %%%%%%%%%%%%%%%%%%%%%%%%
61  %           picture  = TrainValidData{i};
62  %           imwrite(picture,[char(trainLabels(i)) '_' num2str(l) '.jpg'],'jpg
        ');
63  %           imwrite(imgaussfilt(picture, 2),[char(trainLabels(i)) '_' num2str
        (l) '_g.jpg'],'JPG');
64  %           imwrite(imsharpen(picture),[char(trainLabels(i)) '_' num2str(l) '
        _s.jpg'],'JPG');
65  %           imwrite(imrotate(picture,-90),[char(trainLabels(i)) '_' num2str(l
        ) '_r.jpg'],'JPG');
66  %           picture=imnoise(picture,'salt & pepper',0.03);
67  %           imwrite(imrotate(picture,90),[char(trainLabels(i)) '_' num2str(l)
         '_d.jpg'],'JPG')
68
69  %%%%%%%%%%%%  DATA AUGMENTATION WITH NO NOISE (ONLY ROTATIONS) %%%%%%%%%%%%
70
71      picture  = TrainValidData{i};
72  %     picture=imresize(TrainValidData{i},[227, 227]);
73      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) '.jpg'],'jpg');
74      picture  = imrotate(picture,-90);
75      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 'f.jpg'],'jpg');
76      picture  = imrotate(picture,-90);
77      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 's.jpg'],'jpg');
78      picture  = imrotate(picture,-90);
79      imwrite(picture,[char(trainLabels(i)) '_' num2str(l) 't.jpg'],'jpg');
80
81          prevLabel = label;
```

```
        disp(i);
    end
cd ..
trainingLabels = cellstr(trainingLabels);
save trainLabelData trainingLabels % save the labels
```

B.2 Code to train data: Transfer Learning with Alexnet

```
% TITLE: Training CNN using AlexNet transfer Learning *METHOD 2*
% Description: This code train with the data augmented training set using
% AlexNet architecture and freezing only the last three layers weights.
clear all
close all
%%
load('trainLabelData.mat');     % load the label saved from *split data*
all_TrainValid= imageDatastore('TrainValidImages');
all_TrainValid.Labels = categorical(trainingLabels);
% %% SPLITTING TRAINING DATA AND VALIDATION.
[trainImages, validImages] = splitEachLabel(all_TrainValid, 0.857, '
    randomize');
%%
% Resizing train and validation images to 227 x227
trainImages.ReadFcn = @readFunctionTrain;
validImages.ReadFcn = @readFunctionTrain;

save trainvalidImages trainImages validImages %save training data

load('TestData.mat'); % load test data.

% STARTING ALEXNET
```

```
alex = alexnet;
inputSize = alex.Layers(1).InputSize;
layersTransfer = alex.Layers(1:end-3);              %layers transfer
numClasses = numel(categories(trainImages.Labels));% number of clases
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'
        BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
% Parameters to modify before training
opts = trainingOptions('sgdm', 'InitialLearnRate', 1e-3,...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropFactor',0.6,...
    'LearnRateDropPeriod',5,...
    'ValidationData',validImages,...
    'ValidationFrequency',5,...
    'ValidationPatience',Inf,...
    'Plots','training-progress',...
    'MaxEpochs', 20, 'MiniBatchSize', 114,'Shuffle','every-epoch','Verbose'
        ,0,...
    'CheckpointPath','E:\Downloads\kevir_12_Alexnet\DataManagement\weight13
        ');
% load from the check point
 load('convnet_checkpoint__368__2018_06_22__02_52_58.mat'); %from T16
% TRAINING
% myNet = trainNetwork(trainImages, layers, opts); % layer from alexnet.
myNet = trainNetwork(trainImages, net.Layers, opts);% layer from previous
                                                    % learning

%% TESTING AND BUILDING THE CONFUSION MATRIX.
Images = testImages;
Images.ReadFcn = @readFunctionTrain;         % Adapt images 227x227
predictedLabels = classify(myNet, Images);  % Predict
accuracy = mean(predictedLabels == Images.Labels);% Compute accuracy
```

```
54 C = confusionmat(Images.Labels,predictedLabels);  % Build Conf . Matrix
55 figure();plotconfusion(Images.Labels,predictedLabels);% Plot Cong. Matrix
```

B.3 Code to evaluate the accuracy on test data

```
1  % TITLE: Test Evaluation with trained weights.
2  % DESCRIPTION: This code is used to evaluate the accuracy in test data all
3  % along each epoch from the training. For that it is needed the weights
4  % learned per epoch as a paramater.
5  %%
6  clear all
7  close all
8  %%
9  load('TestData.mat');         % load Test Data
10 % load('trainLabelData.mat');
11 load('trainvalidImages.mat');% load train/valid images if required.
12 % load('TrainData.mat');      % load only train data if required.
13 %% EVALUATING THE PERFORMANCE IN TEST,TRAIN AND VALIDATION DATA
14     diras = ['weight13'];
15     eval(['cd ' pwd '\' diras])
16
17     listas=ls('*.mat');
18     accuracyTrain = [];
19     accuracyValid = [];
20     accuracyTest = [];
21 %  trainImages = all_TrainData;
22     for i=1:length(listas(:,1))
23         load(listas(i,:));
24         % Predict per each epoch: Train,Test and validation
25         predictedLabelsTrain = classify(net, trainImages);
26         predictedLabelsValid = classify(net, validImages);
27         predictedLabelsTest = classify(net, testImages);
```

```
28          % Compute the accuracy per each one
29          accuracyTrain(i) = mean(predictedLabelsTrain == trainImages.Labels)
                ;
30          accuracyValid(i) = mean(predictedLabelsValid == validImages.Labels)
                ;
31          accuracyTest(i) = mean(predictedLabelsTest == testImages.Labels);
32
33      end
34      %% PLOT ACCURACIES FROM TEST, TRAIN AND VALID.
35      plot(accuracyTrain,'o');
36      hold on,
37      plot(accuracyValid,'+');
38      hold on,
39      plot(accuracyTest,'*');grid
```

B.4 Code for processing images for method 2.1 and 2.2

```
1  % TITLE: Process the image to detect the center and to crop a small part.
2  % DESCRIPTION: This code is used for the method 2.1 and method 2.2 where is
3  % needed to crop the image from the center in 2.1 into 227x227 and in 2.2
4  % just a small sample that shows only amber with no background at all. Bear
5  % in mind sometimes there is the need to change some parameters in the
6  % processing to work for some images with too much light.
7  %%
8  close all
9  clear all
10 %%
11 for mm=1:12
12     % READ FROM FOLDER
13     if mm>9
```

```
14      diras=['nkk' num2str(mm)];
15      else
16          diras=['nkk0' num2str(mm)];
17      end
18      eval(['cd ' pwd '\' diras])
19
20      listas=ls('*.mat'); %READ .mat extension of the pictures.
21      for i=1:length(listas)
22          load(listas(i,:));
23          picture= Im;
24  %         picture = histeq(picture); % sometimes is needed equalization
25  %% PROCESSING THE IMAGE TO GET ONLY THE OBJECT
26
27  level=graythresh(picture);          % Find the best threshold for the image
28  bi=im2bw(picture,level-level*0.05); % binarize.
29  ni=imcomplement(bi);                % complement of the image binarized
30
31   [lr]=size(ni);
32   ni=[ones(lr(1),1),ni,ones(lr(1),1)];
33   ni=imfill(ni,'holes');             % fill holes
34
35  %   ni=imclearborder(ni,4);         % sometimes is needed to clear borders
36  se7=strel('disk',5);
37  i1=imerode(ni,se7);                 % Erode with a disk structure
38  sel = strel('octagon',6);
39  i1 = imerode(i1,sel);               % Erode with a octagon structure
40  se2 = strel('rectangle',[10,10]);
41  i1 = imerode(i1,se2);               % Erode with a rectangle structure
42  %
43  Ifin2 = i1;
44  [L,n]=bwlabel(Ifin2);               % detect the pixels on the image.
45  %% DETECTING THE CENTRE OF THE IMAGE AND AREA
46  P=regionprops(L,'Centroid','Area','Perimeter','BoundingBox');%
47
48  Centres=[]; Area = []; Perimeter = [];
```

```
% Save all perimeters, areas and centres in a vector
for l=1:n
 Perimeter(l)=P(l).Perimeter;
 Area(l)=P(l).Area;
 Centres=[Centres;P(l).Centroid];
end
% Calculate the circularity of the object
k=0;C=[];
 for k=1:n
    A0=P(k).Area;
    Perimeter=P(k).Perimeter;
    C(k)=Perimeter^2/A0;
 end
%
    [Max,I0] = max(Area); % max area.
    [M,I] = min(C);       % minimum of circularity
    x=P(I).Centroid(1);
    y=P(I).Centroid(2);
    x1 = x-P(I).BoundingBox(1,3)/2 -80; %offset in x 80 to center the amber
    y1 = y-P(I).BoundingBox(1,4)/2 -80; %offset in y 80 to center the amber
%% CROPPING THE IMAGE TO 227X227 OR AN SPECIFIC SIZE.
    height = 226;
    width = 226;
    % limitting the bounding box in case of overpassing the lower limits
    if (y1+height)>480
        y1 = y1- (y1+height-480);

    end
    if (x1+width)>640
        x1 = x1 - (x1+width-640);
    end
    % limitting the bounding box in case of overpassing the upper limits
    if (y1)<0
        y1 = 1;
    end
```

```
84      if (x1)<0
85          x1 = 1;
86      end
87      croppedImage = imcrop(Im, [x1,y1, width, height]); % crop the image.
88      Im=[];
89      imwrite(croppedImage,['Image_' num2str(i) '.jpg'],'JPG') ,
90      end
91      cd ..
92  end %
93  %%
```

Appendix C

YOLO codes

```
% TITLE: Split data into train, test for YOLO.
% DESCRIPTION: This code is used to split the data into train,test and
% validation reading from a single folder where is placed all the images.
% It also creates a .txt file for saving the train and test data
%%
close all
clear all
%% SPLIT TRAINING/TEST
% imagefiles = dir('D:\AmberData\*.jpg');
path =('E:\Downloads\kevir_12_Alexnet\DataManagement\Training\AmberData\');
files = dir('E:\Downloads\kevir_12_Alexnet\DataManagement\Training\
    AmberData\*.jpg');
Q = 1:length(files);
[trainInd,valInd,testInd] = dividerand(Q,0.7,0.3,0.0);

%% TRAINING IMAGES .TXT
ftrain = fopen( 'trainImages.txt','wt');
for i =1:length(trainInd)
    a = files(trainInd(i)).name;

    fprintf(ftrain, '%s\n', ['D:\AmberData\' a]);
```

```
21 end
22 %% VALID IMAGES .TXT
23 fvalid= fopen( 'validImages.txt','wt');
24 for i =1:length(valInd)
25     a = files(valInd(i)).name;
26
27     fprintf(fvalid, '%s\n', ['D:\AmberData\' a]);
28 end
```

Appendix D

Feature codes + non linear classifier codes

```
%TITLE: Code extract features and train an SVM/Ensemble classifier
%DESCRIPTION: this code takes 70% of training set and 30% of testing set to
% use them as a feature extractor and then train SVM or Ensemble classifier
% to use it as a predictor.
%%
clear all
close all
%%
load('trainLabelData.mat');
all_TrainData= imageDatastore('TrainValidImages'); % read images
all_TrainData.Labels = categorical(trainingLabels);% labels
%%
load('TestData.mat');
% %%
testImages.ReadFcn = @readFunctionTrain; % readjust the size.
% CREATE AN ALEXNET CLASSIFIER
alex = alexnet;
inputSize = alex.Layers(1).InputSize;
```

```
19 layersTransfer = alex.Layers(1:end-3);                %layers transfer
20 numClasses = numel(categories(all_TrainData.Labels));% number of clases
21 layer = 'fc7';      % Get the last fully connected layer to get the features
22 % EXTRACT THE FEATURES FROM THE LAST LAYER: TRAIN AND TEST FEATURES
23 featuresTrain = activations(alex,all_TrainData,layer,'OutputAs','rows');
24 featuresTest = activations(alex,testImages,layer,'OutputAs','rows');
25 %
26 YTrain = all_TrainData.Labels;
27 YTest = testImages.Labels;
28
29 % classifier = fitcecoc(featuresTrain,YTrain);  % svm classifier
30 classifier = fitcensemble(featuresTrain,YTrain);% ensemble classifier
31 YPred = predict(classifier,featuresTest);       % predict.
32
33 accuracy = mean(YPred == YTest); % Compute the accuracy
34 %
35 Cmat = confusionmat(YPred, YTest);% compute conf. matrix
36 plotconfusion(YTest,YPred);       % plot the conf. matrix
```